End-to-End Resilience Mechanisms for Network Transport Protocols

BY

Justin P. Rohrer

Submitted to the graduate degree program in Electrical Engineering & Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Chairperson: Prof. James P.G. Sterbenz

Prof. Dr. Bernhard Plattner

Prof. Victor S. Frost

Prof. Joseph B. Evans

Prof. Tyrone Duncan

Dr. David C. Bonner

Date Defended: 11/11/2011

The Dissertation Committee for Justin P. Rohrer
certifies that this is the approved version of the following dissertation:

End-to-End Resilience Mechanisms for Network Transport Protocols

Chairperson: Prof. James P.G. Sterbenz

Date approved: 12/09/2011

Abstract

The universal reliance on and hence the need for resilience in network communications has been well established. Current transport protocols are designed to provide fixed mechanisms for error remediation (if any), using techniques such as ARQ, and offer little or no adaptability to underlying network conditions, or to different sets of application requirements. The ubiquitous TCP transport protocol makes too many assumptions about underlying layers to provide resilient end-to-end service in all network scenarios, especially those which include significant heterogeneity. Additionally the properties of reliability, performability, availability, dependability, and survivability are not explicitly addressed in the design, so there is no support for resilience. This dissertation presents considerations which must be taken in designing new resilience mechanisms for future transport protocols to meet service requirements in the face of various attacks and challenges. The primary mechanisms addressed include diverse end-to-end paths, and multi-mode operation for changing network conditions.

Page left intentionally blank.

Acknowledgments

I would like to acknowledge my advisor, James P.G. Sterbenz for his insight and advice throughout this process. Victor Frost for his feedback and encouragement over the years. Abdul Jabbar, who has always been ready to brainstorm and give insightful comments on this work, as well as developing invaluable MATLAB functions used in the path diversification project, and co-authoring a substantial list of publications with me. Egemen Çetinkaya for his work in categorizing and simulating challenges, as well as his involvement with the ANTP work. Kamakshi Pathapati for her help in writing ns-3 code to simulate the end-to-end ARQ mechanism. Sarvesh Kumar Varatharajan for his work on the ns-2 code for simulating packet size adaptation. I also want to acknowledge Piyush Upadhyay and Ramya Muthyala's work on the cross-layer simulation framework. I would also like to thank the members of my committee for their time in reading this dissertation and the useful comments they have provided. I would like to thank all the staff at the Information and Telecommunication Technology Center for their support through the years. Lastly, a very grateful acknowledgement is due to David Hutchison, Marcus Schöller, Paul Smith, and the other members of the ResiliNets group for their work on the resilient network architecture, which is foundational to this work.

The Path Diversification and ResTP research was supported in part by NSF FIND (Future Internet Design) Program under grant CNS-0626918 (Postmodern Internet Architecture), by NSF grant CNS-1050226 (Multilayer Network Resilience Analysis and Experimentation on GENI), and by the EU FP7 FIRE programme ResumeNet project (grant agreement no. 224619).

For the AeroTP and ANTP research we would like to thank the Test Resource Management Center (TRMC) Test and Evaluation/Science and Technology (T&E/S&T) Program for their support. This work was funded in part by the T&E/S&T Program through the Army PEO STRI Contracting Office, contract number W900KK-09-C-0019 for AeroNP and AeroTP: Aeronautical Network and Transport Protocols for iNET

(ANTP). The Executing Agent and Program Manager work out of the AFFTC. This work was also funded in part by the International Foundation for Telemetering (IFT). We would like to thank Kip Temple and the membership of the iNET working group for discussions that led to this work.

GpENI is funded in part by the US National Science Foundation GENI program (GPO contract no. 9500009441), by the EU FP7 FIRE programme ResumeNet project (grant agreement no. 224619), and in large part by the participating institutions. GpENI is made possible by the involvement of many people in the participating institutions; we particularly acknowledge: At KU: Co-I Joseph Evans; student Egemen Çetinkaya; net/sysadmins Michael Hulet, Wesley Mason. At UMKC: Co-I Baek-Young Choi; students Xuan Liu, Can Kanli, Sean Korzdorfer, Tim Sylvester; net/sysadmins James Schonemann II, George Koffler At UNL: student Mukesh Subedee; net/sysadmins Kent G. Christensen, Dale M. Finkelson (now at Internet2). At KSU: students Ali Sydney and Yunzhao Li (John Sherrell is now at Garmin); net/sysadmins Sam Hays and Richard Becker. At ETH Zürich: student Ajita Gupta. At the University of Cambridge: Co-I Andrew Moore. At Universität Bern: Co-I Torsten Braun. We further acknowledge many other Co-Is, students, and net/sysadmins in the institutions connected to GpENI, listed on the GpENI wiki [1].

Contents

1 Introduction and Motivation **1**

1.1 Communication Networks 2

1.1.1 Challenges to Communication Networks 3

1.1.2 Lack of Explicit Cross-Layering 5

1.2 Problem Statement 6

1.3 Problem Solution 7

1.4 Contributions 7

1.5 Primary Publications 8

1.5.1 Published Journal Articles 8

1.5.2 Journal Articles Under Preparation 9

1.5.3 Conference Papers 9

1.5.4 Technical Reports 13

1.6 Summary 13

2 Background & Related Work **15**

2.1 Cross-Layering Examples 15

2.1.1 Knobs and Dials Formalization 15

2.1.2 Cross-layer Architecture for Simulation 17

2.1.3 Packet Size Adaptation 17

2.2 ResiliNets Architecture 19

2.2.1 Axioms 19

2.2.2 ResiliNets Strategy 19

2.2.3 ResiliNets Principles 20

2.3 Postmodern Internetwork Architecture 20

2.3.1 PoMo Motivation 20

2.3.2 PoMo Architecture . . . 21
2.3.3 PoMo Cross-Layering Framework . . . 22
2.3.4 Transport Layer and PoMo . . . 22
2.4 Transport Protocols . . . 24
2.4.1 General Observations . . . 24
2.5 Diversity . . . 57
2.5.1 Network Survivability . . . 57
2.5.2 Graph Theoretic Diversity Approaches . . . 58
2.5.3 Multipath Routing . . . 59
2.5.4 Multipath Applications . . . 60
2.6 Summary . . . 61

3 Path Diversification **63**
3.1 Motivation for Multipath . . . 65
3.1.1 Terminology . . . 66
3.1.2 Design Goals . . . 67
3.2 Path Diversification Overview . . . 67
3.2.1 Path Diversity . . . 68
3.2.2 Geographic Path Diversity . . . 70
3.2.3 Effective Path Diversity . . . 71
3.2.4 General Path Selection Algorithm . . . 72
3.2.5 Measuring Graph Diversity . . . 73
3.2.6 Terminating Conditions . . . 73
3.3 Evaluation . . . 76
3.3.1 Diversity . . . 79
3.3.2 Dependability . . . 79
3.3.3 Path Stretch . . . 88
3.4 Topology Survivability Comparison . . . 91
3.4.1 Simulation Results . . . 91
3.4.2 Topology Characteristics Survey . . . 97
3.5 Analysis . . . 99
3.5.1 Topology Ranking . . . 99
3.5.2 Compensated Total Graph Diversity . . . 100
3.6 Summary . . . 102

4 Transport Protocol Design **105**

4.1 Resilient Multipath Transport Protocol . . . 107

4.1.1 Protocol Design . . . 107

4.1.2 Mechanism Tradeoffs . . . 109

4.1.3 ResTP Multipath Simulations . . . 109

4.1.4 Summary . . . 116

4.2 Airborne Telemetry Transport . . . 117

4.2.1 Overview of the ANTP Suite . . . 117

4.2.2 Networking Challenges in Airborne Tactical Networks . . . 121

4.2.3 Existing Protocols & Architectures . . . 122

4.2.4 System Architecture . . . 127

4.2.5 AeroTP: TCP-Friendly Transport Protocol . . . 128

4.2.6 Cross-Layer Mechanisms . . . 135

4.2.7 AeroNP: IP-Compatible Network Protocol . . . 136

4.2.8 AeroRP: Location-Aware Adaptive Routing . . . 138

4.3 AeroTP Simulation Model . . . 144

4.3.1 AeroTP Segment Structure . . . 145

4.3.2 AeroTP Operation . . . 146

4.4 Simulation Results . . . 149

4.4.1 AeroTP Connection Establishment . . . 150

4.4.2 Fully-reliable mode performance . . . 151

4.4.3 Quasi-Reliable Mode Characterization . . . 152

4.4.4 Performance Comparison over Lossy Links . . . 157

4.5 Summary . . . 160

5 GpENI Testbed **161**

5.1 Current ANTP Prototyping . . . 161

5.2 Planned ResTP Experiments . . . 162

5.3 GpENI Overview . . . 163

5.3.1 GpENI Programmability and Flexibility . . . 164

5.3.2 GpENI Suitability for ResTP . . . 165

5.4 Topology and Network Infrastructure . . . 166

5.4.1 Midwest US GpENI Core and Optical Backbone 166
5.4.2 GpENI International Topology 167
5.4.3 GpENI Deployment 168
5.5 Node Cluster Architecture 169
5.5.1 GpENI Management and Control 170
5.5.2 Experiment Control 171
5.5.3 Methodology and Cross-Verification 171
5.5.4 Large-Scale Deployment 172
5.6 Summary 173

6 Conclusions and Future Work **175**
6.1 Conclusions 175
6.2 Future Work 176

A AeroTP Protocol Specification **219**
A.1 Architectural Overview 219
A.2 Role of AeroTP 220
A.3 Conventions 221
A.4 AeroTPDU Format 221
A.5 Multiple ACK (MACK) packet format 225
A.6 AeroTP Flow Management 226
A.6.1 Connection Establishment 226
A.6.2 Connection Termination 227
A.7 AeroTP Data Transfer 227
A.7.1 Reliable Mode 228
A.7.2 Quasi-Reliable Mode 229
A.7.3 Unreliable Connection-Oriented Mode 229
A.7.4 Unreliable Connectionless Mode 229
A.7.5 Security Considerations 230

B Tabular Data **233**

C Maps **237**
C.1 Physical Topologies . . . 238
C.2 Logical Topologies . . . 239

D Dynamic Graph Properties **245**

E Flow-Robustness Plots **261**
E.1 Physical Topologies . . . 262
E.1.1 AT&T physical . . . 262
E.1.2 GÉANT2 physical . . . 269
E.1.3 Sprint physical . . . 276
E.2 Logical Topologies . . . 283
E.2.1 AboveNet logical . . . 283
E.2.2 AT&T logical . . . 290
E.2.3 EBONE logical . . . 297
E.2.4 Exodus logical . . . 304
E.2.5 Level-3 logical . . . 311
E.2.6 Sprint logical . . . 318
E.2.7 Telstra logical . . . 325
E.2.8 Tiscali logical . . . 332
E.2.9 Verio logical . . . 339
E.2.10 VSNL logical . . . 346
E.3 Synthetic Topologies . . . 353
E.3.1 Full-mesh . . . 353
E.3.2 Manhattan grid . . . 360
E.3.3 Ring . . . 367
E.3.4 Star . . . 374

Page left intentionally blank.

List of Figures

2.1 packet size adaptation at the transport layer 18
2.2 PoMo layering diagram 23

3.1 Shortest path P_0 and alternatives P_1 and P_2 69
3.2 Geographic diversity: distance d and area A 70
3.3 Sprint logical topology 74
3.4 Sprint physical topology 74
3.5 VSNL logical topology 75
3.6 Level-3 logical topology 75
3.7 Total graph diversity vs. number of paths selected 77
3.8 Total graph diversity vs. effective path diversity threshold 78
3.9 Total graph diversity vs. path-stretch limit 78
3.10 Flow robustness vs. link failure probability for the Sprint logical topology 80
3.11 Flow robustness vs. node failure probability for the Sprint logical topology 81
3.12 Flow robustness vs. node & link failure probability for the Sprint logical topology 82
3.13 Flow robustness vs. node failure probability for the Sprint physical topology 83
3.14 Flow robustness vs. link failure probability for the Level-3 logical topology 83
3.15 Flow robustness vs. node & link failure probability for the VSNL logical topology 84
3.16 Resilience of path diversity for the AT&T physical topology 85
3.17 Resilience of path diversity for the AT&T physical topology 85
3.18 Resilience of path diversity for the AT&T physical topology 86
3.19 Total graph stretch vs. effective path diversity threshold $\lambda = .5$ 87
3.20 Gain vs. effective path diversity threshold where $\lambda = 0.5$ 89
3.21 Gain vs. effective path diversity threshold where $\lambda = 1$ 89

3.22 Gain vs. number of paths requested 90
3.23 Link failure robustness (synthetic) 92
3.24 Node failure robustness (synthetic) 93
3.25 Node & link failure robustness (synthetic) 93
3.26 Link failure robustness (logical) 94
3.27 Node failure robustness (logical) 94
3.28 Node & link failure robustness (logical) 95
3.29 Link failure robustness (physical) 95
3.30 Node failure robustness (physical) 96
3.31 Node & link failure robustness (physical) 96

4.1 ResTP header showing knob and dial fields 107
4.2 Synthetic seven-realm interconnection topology 110
4.3 Internal topology of realm six 111
4.4 simplified AT&T backbone topology 112
4.5 Comparison of flow reliability for synthetic topology 113
4.6 Comparison of flow reliability for AT&T topology 114
4.7 Reduction in performance due to congestion as traffic load increases . . . 115
4.8 Increase in delay as traffic load increases 115
4.9 Dynamic airborne tactical environment 120
4.10 Airborne network protocol architecture 126
4.11 Airborne network protocol functional block diagram 126
4.12 AeroTP connection setup 129
4.13 AeroTP TPDU structure 130
4.14 AeroTP reliable connection transfer mode 132
4.15 AeroTP near-reliable transfer mode 132
4.16 AeroTP quasi-reliable transfer mode 132
4.17 AeroTP unreliable connection transfer mode 133
4.18 AeroTP unreliable datagram transfer mode 133
4.19 AeroNP packet structure 137
4.20 AeroTP data segment structure 145
4.21 AeroTP MACK segment structure 146
4.22 AeroTP connection management 147

4.23 AeroTP state transition diagram 147
4.24 TCP and AeroTP connection establishment delay 151
4.25 Average goodput 152
4.26 Average delay 153
4.27 Cumulative goodput 153
4.28 Cumulative overhead 154
4.29 Average goodput 155
4.30 Average delay 156
4.31 Cumulative goodput 156
4.32 Cumulative overhead 157
4.33 Average goodput 158
4.34 Average delay 158
4.35 Cumulative goodput 159
4.36 Cumulative overhead 159

5.1 GpENI Programmability Layers 164
5.2 GpENI Midwest Topology 167
5.3 GpENI map 168
5.4 GpENI Node Cluster 170

Page left intentionally blank.

List of Tables

3.1 Network Characteristics 76
3.2 Aggregate resilience of path diversity 86
3.3 Network Rank 99
3.4 Compensated TGD 102

4.1 Link stability analysis 123
4.2 Feature Comparison of AeroTP, TP++, UDP, and TCP Variants 128
4.3 Knobs and dials for a telemetry network stack 136
4.4 Feature comparison of AeroRP and other routing protocol categories . . 139
4.5 State & Transition Definitions 148

Page left intentionally blank.

Chapter 1

Introduction and Motivation

The average user now takes for granted communication capabilities that the general population would have considered science fiction a couple of decades ago. Wireless cellular and LAN technologies have become widespread enough for people to rely on their proper functionality for coordinating daily activities. People increasingly choose wireless technologies over wired ones, for example by terminating traditional telephone service and using a cellular phone exclusively. In commercial settings most businesses rely on network services for a number of applications. Internal communication, billing, sales, production control systems, and records all rely on network services and cease to operate in the case of a service outage. In the case of healthcare, the lives of patients can even be at stake if the network does not function as expected. With the increasing consequences of network disruptions, both to the financial and medical health of society, the network becomes an increasingly appealing target for crackers, terrorists, and those involved in information warfare, collectively known as "the bad guys". The US Commission on Critical Infrastructure Protection stated that network attacks have the potential to be catastrophic [2].

The content of this dissertation is as follows: This chapter expands on the arguments just mentioned, as well as mentioning the contributions of this PhD research, and listing

the associated publications, many of which are partially incorporated into later chapters of this work. The second chapter presents background, consisting in part of architectural work contributed by this author and his advisor, as well as a good number of others in the ResiliNets research group. The second chapter also contains the related work, primarily consisting of previous transport and multipath protocols and mechanisms. The third chapter is the core of the dissertation research, consisting of a full development and analysis of *path diversification*. This consists of new metrics and algorithms for choosing end-to-end diverse paths, as well as evaluating network topologies based on their level of diversity. The fourth chapter presents the design of two transport protocols; ResTP, which is based on path diversification combined with a multi-mode reliability architecture, and AeroTP, which is a domain-specific protocol containing many of the features of ResTP, but designed for a single-path environment. Chapter 5 presents the GpENI testbed, which this author has played a large role in implementing and maintaining, and which is now being put to use to testing prototype versions of AeroTP and ResTP. Finally, there are the appendices, which contain data, maps, and numerous plots used to fully analyze the survivability properties of the topologies used in evaluating path diversification, but which require too much space to be included in Chapter 3.

1.1 Communication Networks

The Internet protocol suite had survivability in the face of failures as a design goal [3]. It has also proven its robustness on a large scale, in large part due to the distributed nature of its operational protocols [4]. In spite of this, it quickly becomes apparent that there is a fragility to the performance of any given network application. Unseen perturbations in the network's operational state result in an end-user experience which is far from optimal. Many applications attempt to disguise these lower-level failures, and some are

quite successful, however this is only possible with significant design and programming overhead on a per-application basis. A fundamentally resilient transport protocol could alleviate the need for this overhead by providing selectable survivability levels in a generic manner. The reason for doing this in the end-to-end context as opposed to a lower layer is that the source and/or destination nodes are typically first to be aware of a disruption in service [5], so it makes sense to push control of remediation mechanisms to those hosts.

1.1.1 Challenges to Communication Networks

Challenges to communication networks fall into five categories [6, 7].

1. **Unusual but legitimate traffic:** This includes events such as flash crowds, or sequences of data packets where each packet complies to the service specification but the sequence does not.

2. **Challenges in wireless networks:** These can include episodic connectivity due to noisy wireless channels and node mobility, high bit-error-rates (BER), routing topology changes due to node mobility and highly asymmetric links, resource limitations due to limited power supply, and unpredictably long speed-of-light delay due to changing link lengths and large queue lengths.

3. **Attacks:** Attacks can come in many forms, either against software, i.e. cracking, or through physical hardware damage in the case of natural disasters and acts of war.

4. **Misconfiguration:** Misconfiguration refers to mistakes made in managing the network and other operator errors.

5. **Natural faults:** Natural faults are those which appear over time as network hardware ages and breaks down.

We use the term *resilience* to refer to a networks ability to operate *normally* in the face of these challenges. As defined in [8]:

Definition 1.1.1. *Resilience is the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation.*

By implication, resilience is then a superset of a number of other disciplines that have been studied extensively in the past within various communities.

- **Fault Tolerance:** "We say that a system is fault-tolerant if its programs can be properly executed despite the occurrence of logic faults." [9] quoted in [10]. "The ability of a functional entity to mask or mitigate the impact of faults on its specified operation." [11]

- **Dependability:** "Dependability is that property of a [computing] system which allows reliance to be justifiably placed on the service it delivers" [12]. "Dependability includes both reliability and availability." [13, 14]

  - **Reliability:** "The probability that an entity (unit) will complete its intended mission (i.e., perform a specific function) as required over a specified period of time in its intended environment (or stated conditions)." [11]

  - **Availability:** "The proportion of the operating time in which an entity meets its in-service functional and performance requirements in its intended environment." [11]

- **Survivability:** "The capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents." [15, 16]

- **Robustness:** "The ability of a system to maintain specified features when subject to assemblages of perturbations either internal or external." [17]

- **Performability:** "The property of a computer system such that it delivers performance required by the service, as described by QoS (quality of service) measures." [6] "Performance refers to how effectively and efficiently a system delivers a specified service, presuming it is delivered correctly." [18]

- **Disruption Tolerance:** "The ability of a system to tolerate disruptions in connectivity among its components. Disruption tolerance is a superset of tolerance of the environmental challenges: weak and episodic, channel connectivity, mobility, delay tolerance, as well as challenges due to power and energy constraints." [6, 19]

Each of these disciplines contributes to the overall resilience of a system.

1.1.2 Lack of Explicit Cross-Layering

Current general purpose network stacks observe strict layering, due to a desire to separate concerns and maintain simplicity in the network core (e.g. anything over IP over anything). A common representation of these layers is a strict interpretation of the seven-layer OSI Model [20]. The lack of information exchange that this causes leads to invalid assumptions being made, particularly at the the transport layer (TCP assumes congestion for congestion, corruption, and long delay).

That being said, there are some forms of cross-lyering which are widely accepted and used. One of these is the TCP congestion control mechanism which uses packet-loss events to trigger a reduction in flow and prevent congestion. Because this is an implicit signaling it is not specific enough to indicate the cause of the packet loss, so TCP assumes congestion as a safe default. These assumptions can lead to poor performance and gross inefficiency, not because of limitations in the network, but because the transport layer does not correctly adapt to the current network conditions.

Cross-layering is not an end in itself, but it is an essential support capability in order for the mechanisms discussed in the previous section to operate effectively. It uses dials to provide information to higher layers, and knobs to influence the operation of lower layers [21, 22]. Making proper use of these allows the transport layer to adapt to the operational condition of the underlying network, as well as modifying its mode of operation based on the requirements of the applications above it. These cross-layer knobs and dials enable the selection and tuning of specific resilience mechanisms that are appropriate for the current network state.

1.2 Problem Statement

Current transport protocols are designed to provide fixed mechanisms for error remediation (if any), using techniques such as ARQ, and offer little or no adaptability to underlying network conditions, or to different sets of application requirements. In the case of TCP, when a connection times out or too many packets are dropped it simply closes the connection. UDP is not even aware of packet losses that occur. Additionally the properties of reliability, availability, dependability, and survivability are not explicitly addressed in the design, so there is no support for resilience.

Thesis Statement:

> End-to-end communication with resilience as an inherent design property is necessary to meet specified service requirements in the face of various attacks and challenges. Diversity is an essential characteristic of network topologies that enables enhanced resilience at the end-to-end layer.

1.3 Problem Solution

By applying ResiliNets principles [8] in the end-to-end (E2E) context, we have decided on the following research thrusts to the solution: End-to-end diverse multipath, and adaptable reliability paradigms. These components are distinct, but also interdependent. Each component is then broken down into the following two phases: (i) identify the set of alternative mechanisms necessary to support resilience, and (ii) model and compare the mechanisms in a simulation environment to identify tradeoffs between them. Subsequently, these components are integrated into one adaptable protocol, and evaluated through simulation.

1.4 Contributions

They key contributions of this research consist of the path diversification method and associated metrics that allow prediction of topology survivability and the ResTP resilient transport protocol architecture, which are presented in Chapters 3 and 4 respectively. The research efforts in these areas has resulted in a number of supporting contributions as well. The primary and supporting contributions of this dissertation are as follows:

- Primary Contributions
  - Path diversification algorithm
  - Prediction of network survivability using path diversification
  - ResTP multipath protocol design
- Supporting Contributions
  - Path diversification implementation in MATLAB

- Distributed path diversification implementation for parallel processing and protocol use
- Comparative analysis of path diversification with traditional graph theory metrics (betweenness, degree, k-connectedness)
- Understanding tradeoffs between protocol reliability mechanisms
- Comparative analysis of protocol reliability mechanisms under adverse conditions
- ANTP cross-layer protocol stack architecture
- AeroTP RFC-style specification
- AeroTP simulation model
- AeroTP reference implementation
- Building and managing the GpENI testbed

1.5 Primary Publications

The research presented in this dissertation has resulted in a number of publications, including the following.

1.5.1 Published Journal Articles

1. **Justin P. Rohrer**, Abdul Jabbar, Egemen K. Çetinkaya, Erik Perrins, and James P.G. Sterbenz. Highly-dynamic cross-layered aeronautical network architecture. *IEEE Transactions on Aerospace and Electronic Systems (TAES)*, 47(4):2742–2765, October 2011.

2. Abdul Jabbar, **Justin P. Rohrer**, Victor S. Frost, and James P. G. Sterbenz. Survivable millimeter-wave mesh networks. *Computer Communications (COMCOM)*, 34(16):1942–1955, October 2011.

3. James P.G. Sterbenz, Egemen K. Çetinkaya, Mahmood A. Hameed, Abdul Jabbar, Qian Shi, and **Justin P. Rohrer**. Evaluation of network resilience, survivability, and disruption tolerance: Analysis, topology generation, simulation, and experimentation (invited paper). *Springer Telecommunication Systems*, 2011. (accepted March 2011).

4. James P. G. Sterbenz, David Hutchison, Egemen K. Çetinkaya, Abdul Jabbar, **Justin P. Rohrer**, Marcus Schöller, and Paul Smith. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks: Special Issue on Resilient and Survivable Networks (COMNET)*, 54(8):1245–1265, June 2010.

1.5.2 Journal Articles Under Preparation

1. **Justin P. Rohrer** and James P.G. Sterbenz. Path Diversification. *Springer Telecommunication Systems*, 2012.

2. **Justin P. Rohrer** and James P.G. Sterbenz. Reliability Paradigms in Transport Protocols: a Survey. *IEEE Communications Surveys & Tutorials*, 2012.

1.5.3 Conference Papers

1. **Justin P. Rohrer**, Egemen K. Cetinkaya, Hemmanth Narra, Dan Broyles, Kevin Peters, and James P. G. Sterbenz. AeroRP performance in highly-dynamic airborne networks using 3D gauss-markov mobility model. In *Proceedings of the IEEE Mil-*

itary Communications Conference (MILCOM), Baltimore, MD, USA, November 7–10 2011.

2. Mohammed AL-Enazi, Santosh Ajith Gogi, Dongsheng Zhang, Egemen K. Çetinkaya, **Justin P. Rohrer**, and James P. G. Sterbenz. ANTP protocol suite software implementation architecture in python. In *International Telemetering Conference (ITC)*, Las Vegas, NV, October 2011.

3. Kamakshi Sirisha Pathapati, Anh Nguyen, **Justin P. Rohrer**, and James P.G. Sterbenz. Performance analysis of the AeroTP transport protocol for highly-dynamic airborne telemetry networks. In *Proceedings of the International Telemetering Conference (ITC)*, Las Vegas, NV, October 2011, **awarded best graduate-student paper**.

4. **Justin P. Rohrer** and James P. G. Sterbenz. Predicting topology survivability using path diversity. In *Proceedings of the IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, pages 95–101, Budapest, Hungary, October 5–7 2011, pp. 95-101, **nominated for best paper**.

5. **Justin P. Rohrer**, Egemen K. Çetinkaya, and James P.G. Sterbenz. Resilience experiments in the GpENI programmable future internet testbed. In *Proceedings of the 11th Würzburg Workshop on IP: Joint ITG and Euro-NF Workshop "Visions of Future Generation Networks" (EuroView2011)*, August 2011.

6. **Justin P. Rohrer**, Egemen K. Çetinkaya, and James P. G. Sterbenz. Progress and challenges in large-scale future internet experimentation using the GpENI programmable testbed. In *The 6th ACM International Conference on Future Internet Technologies (CFI)*, pages 46–49, Seoul, Korea, June 2011.

7. Hemanth Narra, Yufei Cheng, Egemen K. Çetinkaya, **Justin P. Rohrer**, and James P.G. Sterbenz. Destination-sequenced distance vector (DSDV) routing protocol implementation in ns-3. In *Proceedings of the ICST SIMUTools Workshop on ns-3 (WNS3)*, Barcelona, Spain, March 2011.

8. James P.G. Sterbenz, Egemen K. Çetinkaya, Mahmood A. Hameed, Abdul Jabbar, and **Justin P. Rohrer**. Modelling and analysis of network resilience (invited paper). In *Proceedings of the Third IEEE International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–10, Bangalore, India, January 2011.

9. **Justin P. Rohrer**, Abdul Jabbar, Egemen K. Çetinkaya, and James P.G. Sterbenz. Airborne telemetry networks: Challenges and solutions in the ANTP suite. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 74–79, San Jose, CA, USA, November 2010.

10. Kamakshi Sirisha Pathapati, **Justin P. Rohrer**, and James P. G. Sterbenz. Edge-to-edge ARQ: Transport-layer reliability for airborne telemetry networks. In *Proceedings of the International Telemetering Conference (ITC)*, San Diego, CA, October 2010.

11. James P. G. Sterbenz, Deep Medhi, Byrav Ramamurthy, Caterina Scoglio, David Hutchison, Bernhard Plattner, Tricha Anjali, Andrew Scott, Cort Buffington, Gregory E. Monaco, Don Gruenbacher, Rick McMullen, **Justin P. Rohrer**, John Sherrell, Pragatheeswaran Angu, Ramkumar Cherukuri, Haiyang Qian, and Nidhi Tare. The Great plains Environment for Network Innovation (GpENI): A programmable testbed for future internet architecture research. In *Proceedings of the 6th International Conference on Testbeds and Research Infrastructures for the*

Development of Networks & Communities (TridentCom), pages 428–441, Berlin, Germany, May 18–20 2010.

12. **Justin P. Rohrer**, Ramya Naidu, and James P. G. Sterbenz. Multipath at the transport layer: An end-to-end resilience mechanism. In *Proceedings of the IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, pages 1–7, St. Petersburg, Russia, October 2009.

13. **Justin P. Rohrer** and James P. G. Sterbenz. Performance and disruption tolerance of transport protocols for airborne telemetry networks. In *Proceedings of the International Telemetering Conference (ITC) 2009*, Las Vegas, NV, October 2009.

14. **Justin P. Rohrer**, Abdul Jabbar, and James P. G. Sterbenz. Path diversification: A multipath resilience mechanism. In *Proceedings of the IEEE 7th International Workshop on the Design of Reliable Communication Networks (DRCN)*, pages 343–351, Washington, DC, USA, October 2009.

15. Abdul Jabbar, **Justin P. Rohrer**, Andrew Oberthaler, Egemen K. Çetinkaya, Victor Frost, and James P. G. Sterbenz. Performance comparison of weather disruption-tolerant cross-layer routing algorithms. In *Proc. IEEE INFOCOM 2009. The 28th Conference on Computer Communications*, pages 1143–1151, April 2009.

16. **Justin P. Rohrer**, Abdul Jabbar, Erik Perrins, and James P. G. Sterbenz. Cross-layer architectural framework for highly-mobile multihop airborne telemetry networks. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 1–9, San Diego, CA, USA, November 2008.

17. **Justin P. Rohrer**, Erik Perrins, and James P. G. Sterbenz. End-to-end disruption-tolerant transport protocol issues and design for airborne telemetry networks. In

Proceedings of the International Telemetering Conference, San Diego, CA, October 27–30 2008, **awarded best paper**.

1.5.4 Technical Reports

1. James P.G. Sterbenz, **Justin P. Rohrer**, and Egemen K. Çetinkaya. Multilayer network resilience analysis and experimentation on GENI. ITTC Technical Report ITTC-FY2011-TR-61349-01, The University of Kansas, Lawrence, KS, July 2010.

1.6 Summary

In this chapter we have given an overview of the challenges that we will be addressing in this dissertation, as well as summarizing the methods we have used to address them. In the next chapter we will present the background which gives context to this work, and related work consisting of a survey of transport protocols and multipath mechanisms.

Page left intentionally blank.

Chapter 2

Background & Related Work

In this chapter we look first at a number of existing efforts that form the background of this work, including the benefits of cross-layering (Section 2.1), the ResiliNets architecture (Section 2.2), and the postmodern internetwork architecture (Section 2.3). Secondly we examine the related work, including a broad ranch of general purpose and specialized transport protocols (Section 2.4) from which we gain insight into the range of mechanisms which have previously been used to address specific challenges in end-to-end reliability. Lastly in Section 2.5 we examine existing diversity research, including multipath routing protocols.

2.1 Cross-Layering Examples

This section gives a brief overview of of several efforts we have made to formalize cross-layering, aspects of which are applied in this work.

2.1.1 Knobs and Dials Formalization

When evaluating decisions regarding locating functionality at a given layer we are primarily evaluating the tradeoffs involved. Design complexity is one of the primary concerns

in cross-layering, so we evaluate the tradeoff of increased complexity compared to the increase in resilience. We (The ResiliNets group, collaboratively) have begun to formalize the representation of knobs and dials as follows.

The set of all knobs

$$\mathbb{K} = \mathbf{K} \cup \mathbf{k} \tag{2.1}$$

is the union of out-of-band $\mathbf{K}$ and in-band $\mathbf{k}$. The set of all dials

$$\mathbb{D} = \mathbf{D} \cup \mathbf{d} \tag{2.2}$$

is the union of out-of-band $\mathbf{D}$ and in-band $\mathbf{d}$. Knobs and dials are defined on the boundary between layers L_i and L_j where i and j are either numbers, e.g. {1, 1.5, 2, 2.5, 3, 4, 7, 8} or layer designators, e.g {HBH, net, PoMo, E2E, app}. An individual knob or dial between layers L_i and L_j is then $K_{i \to j}(\text{desc})$ where 'desc' is a descriptor, e.g. BER. Therefore the set of all out-of-band knobs and dials between layers i and j

$$\mathbf{K} = \cup_{\forall K \in \mathbf{K}} K_{i \to j} \tag{2.3}$$

represents a vertical relationship when $i \neq j$ and represents a horizontal relationship when $i = j$.

To carry this further, we can fully represent a layer n protocol instance at time t in terms of its knobs and dials, as well as its *state* $s(t)$ and *context* $c_n(t)$. For L_n, we can define state with respect to time as:

$$\mathrm{s}(t+1) = f(\mathbb{K}_{n+1 \to n}, \mathbb{D}_{n \leftarrow n-1}, \mathrm{s}(t), \mathrm{c}_n) \tag{2.4}$$

where f is a function specific to the internal algorithm of that particular protocol.

We apply this formalism to mechanisms such as explicit cross-layer support for error control that allow the network layer to notify the end-to-end layer of the cause of data

loss. ELN (explicit loss notification) notifies the transport layer of loss due to corruption, ECN (explicit congestion notification) notifies it that loss was due to congestion, EON (explicit outage notification) not only notifies it of loss but that there is an outage in the path, and EDN (explicit delay notification) notifies the transport layer that data has been delayed but not actually lost. In addition to ELN, we could also use cross-layer notification to indicate that some corruption was experienced but it was recoverable using FEC.

2.1.2 Cross-layer Architecture for Simulation

We have begun implementing the framework needed to support cross-layering as discussed in Section 2.3.3 in the ns-2 simulator. The ns-2 simulator does not transfer a payload or header bits in its simulated data packets, so it is impossible to send control packets for cross-layering within the existing packet model. Instead we create global data structures within the simulation framework in which to store the packet contents, and then do a lookup based on the global identifier of each packet to find the contents of the packet. For the time being we only do out-of-band cross-layer messaging, but we hope to be able to add in-band messaging in the future.[1]

2.1.3 Packet Size Adaptation

To demonstrate the performance gains enabled by the architecture in Section 2.1.2 we simulated a transport protocol that adapts the TPDU size depending on the BER observed at the link layer. In wired networks there is generally a performance benefit to sending large packets, thus requiring fewer packets to send a given amount of data and incurring less overhead in the form of packet headers. As the number of bit-errors in-

[1]Work discussed in Section 2.1.2 was performed jointly with Sarvesh Kumar Varatharajan, Piyush Upadhyay, and Ramya Muthyala; future work will be transitioned to the ns-3 simulator.

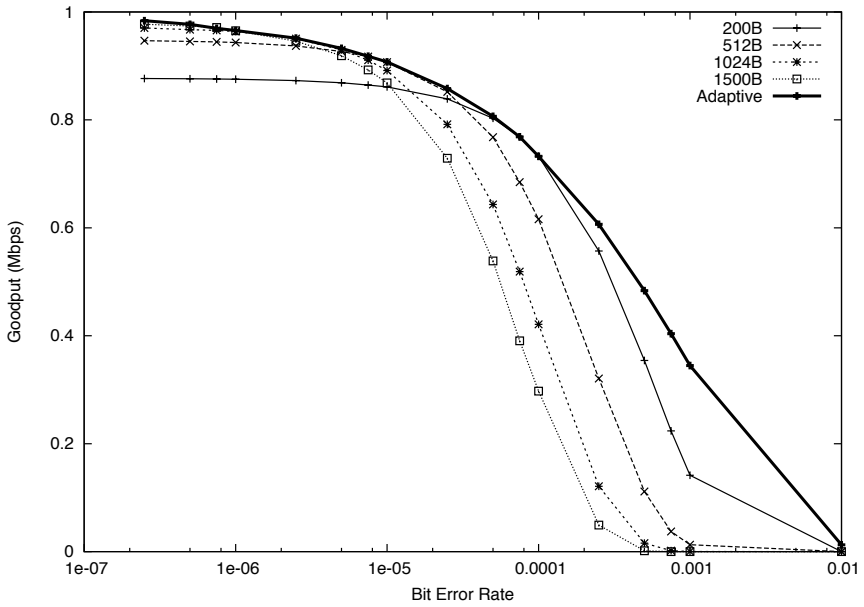

Figure 2.1: packet size adaptation at the transport layer

creases, such as in wireless networks, the probability of packet loss due to corruption increases and large packets are both more likely to be corrupted and incur a greater penalty because more data is lost as a result of each bit error. The result of this situation is that there is an optimal packet size given a particular header length and channel bit-error rate. We were able to use the algorithm from [23] to simulate an example of this in ns-2 in which we achieve optimal goodput by varying the packet size depending on the BER of the underlying links. The simulation consisted of a 1 Mb/s link with constant bit rate traffic being transferred and varying average bit-error rates.

Figure 2.1 shows the goodput achieved with packet sizes fixed at 200, 512, 1024, and 1500 bytes, as well as adaptive packet size that varies as the BER on the link changes. From the plot we can see that the adaptive transport layer achieves the maximum goodput in

all cases, composing the envelope of all the fixed-size curves.[2]

2.2 ResiliNets Architecture

The ResiliNets group [6] has established an architecture for designing resilient networks. This consists of a set of foundational axioms, a strategy for implementing resilience, and as set of supporting principles [8].[3]

2.2.1 Axioms

The ResiliNets Axioms are a small set of fundamental assumptions that need to be taken into consideration when designing a resilient system. These include that faults and challenges are inevitable, that normal operations must be understood before deviations from the norm can be detected, that challenges of some type are going to occur, and that the system needs to respond to these challenges.

2.2.2 ResiliNets Strategy

The ResiliNets strategy consists of two phases D^2R^2+DR, which can be thought of as the real-time operations, and the long-term operations. The real-time phase D^2R^2 consists of defending against challenges, detecting errors (which are caused by challenges that activate faults), remediating to do the best possible given the conditions of the challenge, and lastly recovering to the normal operational state once the challenge has passed. In this work on resilient end-to-end transport, we are primary addressing the defense and remediation aspects of the strategy. Defense is addressed through mechanisms which

[2]Work discussed in Section 2.1.3 performed jointly with Sarvesh Kumar Varatharajan.

[3]The article cited here was written collaboratively by several members of the ResiliNets research group, including the author of this dissertation. It provides much more extensive background on the understanding and development of resilient systems. This document presents only a small subset of that thinking, which is specifically applied to the end-to-end layer.

use some form of redundancy to proactively guard agains data loss. Remediation is addressed by those that retransmit or fail-over to diverse alternative options in the event that data-loss occurs.

2.2.3 ResiliNets Principles

The ResiliNets architectural principles support the strategy as a set of best-practice concepts to apply when designing a resilient system. In this work we identify a subset of the principles that are particularly relevant to the end-to-end layer and apply them to the design of a resilient transport protocol.

2.3 Postmodern Internetwork Architecture

The Postmodern Internetwork Architecture (PoMo) project [24] is funded by the National Science Foundation (NSF) [25] under the Future Internet Design (FIND) program [26]. PoMo is a greenfield architecture for the future Internet that seeks to separate policy implementation from packet forwarding mechanisms and support heterogeneous internetworking, by explicitly providing a realm interconnection layer which provides translation services at mechanism, trust, and policy boundaries.

2.3.1 PoMo Motivation

The current Internet is constructed of an increasingly heterogeneous mix of underlying technologies, all hidden under the IP blanket and assumed by upper layers to function as a stable and well-connected homogeneous environment. The universal dominance of IP, as well as the TCP and UDP transport protocols has led to extreme difficulty in network and transport layer innovation, since applications expect only to operate over these few protocols and are not designed to operate flexibly across a variety of protocols

and technologies. Many researchers have proposed solutions to recognized problems in the current Internet architecture, however the community requirement for backwards compatibility makes widespread adoption virtually impossible. For this reason PoMo is designed as a greenfield architecture.

Heterogeneity, which is a natural form of diversity and should therefore improve resilience, is instead an obstacle to connectivity and performance in the current Internet. One example where the effects of this can be clearly seen is in cellular data networks. Attempting to extend the IP Internet to this wireless realm has resulted in numerous application hacks and limitations [27,28], massive overhead, and a protocol stack with multiple overlays so complex (15 layers deep in some places) it is very difficult to understand [29].

2.3.2 PoMo Architecture

The intention of PoMo is to add a thin internetwork layer, substantially similar to the *original* purpose of IP, to enable the interconnection of heterogeneous realms. Heterogeneous realms are those that employ diverse mechanisms (including protocols stacks and addressing paradigms) internally. The primary responsibility of the PoMo layer is to handle the resolution of policy tussles explicitly, rather than manipulating the packet-forwarding path to accomplish this as is currently done with BGP. The primary principles of PoMo are "(i) strict separation of concerns, and (ii) inclusion of explicit mechanisms in support of all foreseeable policies influencing network-layer behavior" [24].

Based on these two principles, the PoMo architecture implements several unique characteristics. First, the realm-level path is differentiated from hop-by-hop forwarding, and the user is allowed to determine the real-level path via a source routing construct, thus giving greater control to users, while maintaining the service providers ability to control intra-realm forwarding control for traffic engineering. At the same time this realm-level

path is captured and delivered to the destination, providing an accountability mechanism for tracking down spammers and DOS initiators. Lastly, PoMo explicitly supports cross-layer *knobs* and *dials* to enable the communication of network state information (e.g. error rates, congestion, path availability) upwards to the user, and the downward influence of policy decisions (i.e. service requirements) from the user to the network.

2.3.3 PoMo Cross-Layering Framework

In the context of the Postmodern Internetwork Architecture presented in Section 2.3, we are developing a model for a transport protocol appropriate for resilient E2E communication over a heterogeneous internetwork. It communicates directly with the PoMo layer at realm boundaries. A depiction of this layering is shown in Figure 2.2. We are showing three realms with diverse mechanisms interconnected to form a heterogeneous internetwork. In this figure the dashed lines represent the logical communication that is taking place at each layer, while the solid lines represent the actual data transfer both between layers and over physical wires. The different network layers cannot communicate across the realm boundaries, so the PoMo internetwork layer (L3.5) provides the necessary translation.

2.3.4 Transport Layer and PoMo

PoMo is not an end-to-end protocol nor does it define constraints on the implementation of one, however several aspects of the architecture, and the *knobs and dials* in particular (Figure 2.2), give us an opportunity to design a resilient transport-layer protocol utilizing a much greater level of interaction with the underlying network than is possible in the current Internet. This interaction allows us to overcome the limitations imposed by the lack of cross-layering discussed in Section 1.1.2.

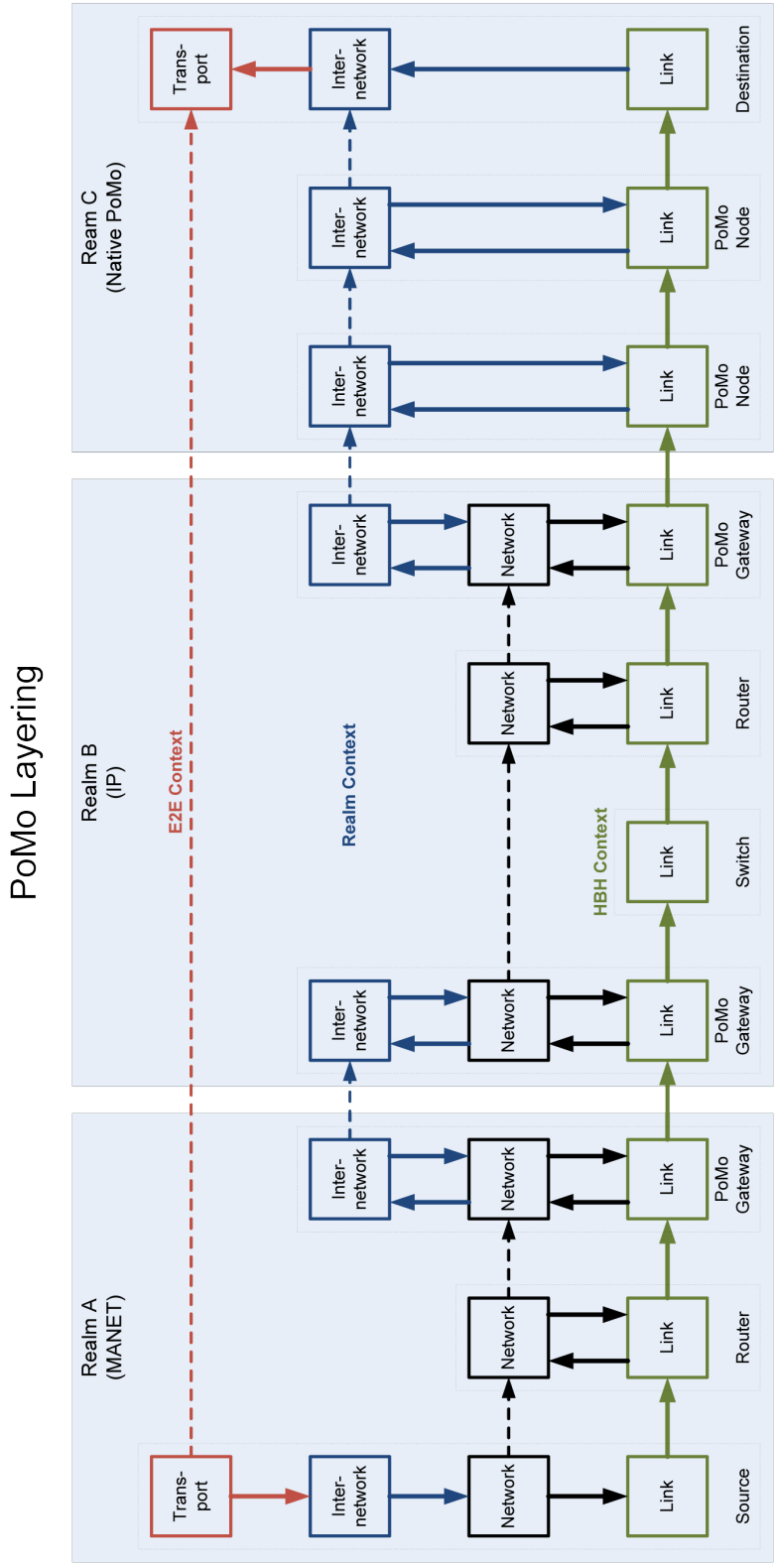

Figure 2.2: PoMo layering diagram

2.4 Transport Protocols

In this section we present a comprehensive survey of transport protocols and their intended use, specifically noting the reliability mechanisms employed. We also include protocols without explicit reliability mechanisms, since *no reliability* is a legitimate paradigm choice for some applications and networks.

A number of surveys have been done addressing domain-specific transport protocols such as ad-hoc networks [30], high-speed networks [31], space links [32], wireless sensor networks [33, 34], and general wireless networks [35]. To our knowledge this is the first work to survey reliability mechanisms across all these challenged domains, as well as presenting the early protocols from which most of the modern ones are derived.

2.4.1 General Observations

A significant portion of transport research has focused on challenged environments. It is not difficult to design a transport protocol for well-connected reliable networks. In the ideal case, UDP would be sufficient, because the network would be completely reliable and not drop any packets. After the early years of transport protocol development, research has focused on satellite and sensor network environments, because they present the greatest challenges to maintaining end-to-end reliability.

We have chosen to organize this section chronologically for ease in conveying to the reader the dependence of later protocol designs on earlier work. We note that this is a rough chronology, in that some protocols are best known through a source that is not the earliest in which they appeared. A prime example of this are protocols defined in RFCs, but that appeared in earlier literature.

1970–1989

TCP: TCP is tightly coupled to IP, relying on it to provide a unified edge-to-edge datagram service, independent of underlying network technologies. It is a reliable byte-stream protocol, adding both end-to-end acknowledgements (in its original form) [36,37], as well as congestion control (introduced later) [38,38,39] to prevent congestion collapse in the Internet. TCP is connection-oriented, using a three-way handshake to establish connections, and two two-way handshakes to terminate the connection. It uses sequence numbers to enable the detection of lost segments, and uses a weak 16-bit (non-CRC) checksum to detect errors. The TCP receiver advertises a window size that the sender uses to regulate the amount of data sent out in a burst and prevent overwhelming the receivers buffer. Due to its wide adoption, more versions of TCP have been proposed in the literature than of any other transport protocol of which we are aware. Selected TCP modifications will be described later in the section, in the order in which they were introduced. Investigating the early versions of TCP is somewhat convoluted due to the fact that they were implementations based on collated theoretical mechanisms proposed in the literature, and refereed to simply as "TCP", and it was not until Kevin Fall and Sally Floyd's paper years later [40] that the names TCP-Tahoe, TCP-Reno, (Tahoe and Reno being the release names of the BSD version in which their implementation first appeared) and TCP-SACK came into being. Later it became common for researchers to provide a new name for their modified TCP protocol (e.g. TCP-Vegas [41]).

Delta-t: Delta-t was developed and used for a number of years at the Lawrence Livermore labs. Like initial versions of TCP it offers a reliable byte-stream end-to-end service without congestion control. Unlike TCP, Delta-t uses timer-based connection management. Connection setup is explicit on the receipt of the first data segment, and the connection is closed when a timeout occurs following the last data exchange. Unacknowledged data

is retransmitted after a timeout and reordering is handled by the receiver [42–45].

UDP: UDP [46] falls into the "general purpose" category and is the simplest of the widely used transport protocols. It is stateless and does not offer any service beyond application multiplexing on top of IP. It simply sends packets with no assurance or notification of correct delivery.

Datakit: The Datakit network assumes circuit-oriented network that delivers segments in order and error-free. Its only function with respect to reliability is to retransmit lost packets. Protocol operations are controlled by the transmitter, so that the receiver operation can be simplified to reduce the processing required on the receiver side. Other interesting features include the use of a 9-bit byte. The 9^{th} bit is used to indicate whether that byte is data or control information, so that the control commands to the receiver may be interspersed with the data stream. Control bytes can indicate that the receiver should acknowledge the data, and that a set of data bytes should be passed to the transport layer as a "block", thus enabling fragmentation into smaller cells by the lower layers without requiring additional overhead for reassembly. The transmitter can also specify that the receiver should report losses, so that they can be retransmitted [47, 48].

RDP: The reliable data protocol is a reliable transport protocol based on TCP that uses window-based flow control *without* congestion control. It also makes in-order delivery optional, so that the application can enable or disable it depending on its requirements. These changes from the TCP design are intended to reduce both the implementation complexity, and minimum resource requirements (turning off reordering reduces buffer space required at the receiver) [49, 50].

APPN: Advanced peer-to-peer networking, while not specifically a transport protocol, does define transport-layer functions built on top of IBM's SNA architecture. APPN includes functions to setup and release end-to-end connections in a virtual-circuit envi-

ronment, and assumes that full reliability is provided by the data-link layer. Therefore it does not address error handling at the end-to-end layer [51–53].

VMTP: The versatile message transfer protocol was designed to support transactional traffic, specifically remote procedure calls (RPC's) used in a distributed operating system, which require low latency, but transfer only small amounts of data. Additionally, VMTP provides a streaming mode that provides both rate and reliability, and uses a selective-repeat algorithm [54–56].

NETBLT: The network block transfer protocol [57–60] is an application specific protocol designed for transporting large blocks of data with high throughput [31]. It was designed to operate efficiently even over long-delay links such as those found in satellite networks, and use the IP network layer. NETBLT connections are unidirectional, and may only be released by the sender. It addresses flow control using both window and rate-control mechanisms. NETBLT also performs error-control using an optimized selective repeat mechanism which can fill multiple holes using a single retransmission request message.

DECbit: DECbit is an explicit congestion notification mechanism. It consists of a single bit in the transport header, set by a congested hop on packets traveling in the forward direction, and relayed to the sender via acknowledgment packets. Based on this decision bit, the sender can then reduce its sending rate based on explicit knowledge, as opposed to implicitly assuming congestion due to a packet loss as in done in TCP [61].

TCP-Tahoe: TCP-Tahoe is the 4.3 BSD Tahoe version of TCP implemented in 1988 and based on Van Jacobson's congestion-control theory [62], but did not receive its name until Fall & Floyd's 1996 paper [40]. Its distinctive feature is the addition of the basic congestion control strategy that is used today to the basic TCP functionality. This includes the maintenance of a *congestion window* that limits the number of unacknowledged packets in flight, in addition to the limit previously imposed by the receive window. It also

includes the *slow start* state to rapidly ramp up the sending rate when a connection is initialized. Thirdly, Tahoe specifies a *congestion avoidance* state, in which the sending rate is gradually increased until a loss occurs. When the receiver in TCP-Tahoe receives a mis-ordered segment, it does not acknowledge it, instead it sends a duplicate acknowledgement for the last in-order packet received. After three duplicate acknowledgements the sender performs a *Fast Retransmit* by sending the first unacknowledged packet and resets its congestion window to one segment beginning the slow-start process over again.

ISO-TP0–4: The OSI transport protocol suite (ISO-TP) is functionally partitioned into five distinct protocols TP0 through TP4 that are meant to provide increasing levels of reliability, TP0 being comparable to UDP and TP4 comparable to TCP. The user or application is required to select the correct TP variant to meet its requirements given the characteristics of the underlying network [63–65]. OSI/TP4 is a connection-oriented protocol, designed to provide reliability on top of an unreliable network service. It also has a number of mechanisms for negotiating quality-of-service parameters including required throughput, priority, delay, and error-rate. Similarly to Delta-t, error recovery happens based on timeouts alone, there is no negative acknowledgement mechanism in the protocol.

CARD: Congestion avoidance using round-trip delay was proposed as an alternative to DECbit (by the same authors), which unlike DECbit does not require participation of the network layer. Instead CARD relies on the RTT to set the window size. There are 4 cases when using the CARD algorithm. If the window size is increased and the RTT increases, decrease the window size by $\frac{1}{8}$. If the window size is increased and the RTT does not increase, increase the windows size by 1 MSS. If the window size is decreased and the RTT increases, increase the window size by 1 MSS. Lastly if the window size is decreased and the RTT decreases, decrease the window size by $\frac{1}{8}$. The result is a window size which continually oscillates around it optimal value, assuming congestion is the only

cause of RTT change [66].

1990–1999

TCP-Reno: TCP-Reno is an optimized implementation of TCP-Tahoe and first appeared in 4.3 BSD Reno in 1990, but like TCP-Tahoe, it did not receive its name until the Fall, Floyd 1996 paper [40]. It added an additional state called *Fast Recovery* that occurs after a Fast Retransmit event, and instead of resetting the congestion window to 1 segment, it instead sets it to half of its current value. This was shown to be sufficient reduction in sending rate to prevent congestion collapse, while reducing the burstyness of the traffic flow and thereby increasing the overall utilization of of the network.

ALTP-OT: The application-oriented lightweight transport protocol is part of the Axon high-speed communication architecture for distributed applications [67]. Flow-control in this environment is rate-based, and bandwidth is reserved during connection setup, making congestion control unnecessary. Error correction is handled with selective retransmission that is either timer-based or tightly coupled to the demands of the application, meaning that a missing packet containing data that is not referenced by the application may never be retransmitted [68].

Tri-S: Slow start and search is an end-to-end congestion control algorithm that operates by evaluating the marginal increase in throughput gained by each MSS added to the window size. If the marginal increase drops below half the throughput when the window size was 1 MSS, the window size is decreased. Tri-S uses additive increase and additive increase, assuming fairness to have been established during session startup [69].

DUAL: Crowcroft's dual adjustment scheme is designed to avoid congestion while inducing less oscillation in aggregate network traffic than Jacobson's AIMD algorithm found in TCP-Tahoe. It works by comparing the RTT against the average of previously observed

minimum and maximum RTTs. If the current RTT is above this average the congestion window is reduced by $\frac{1}{8}$, otherwise it is increased by 1 MSS. [70].

XTP: The express transfer protocol is designed to provide multiple service types, including bulk data transfer, real-time datagrams, and multicast. It was also explicitly designed for ease of implementation using VLSI. It is a unified transport and network layer, and uses a combination of rate-control, selective retransmission, and implicit connection establishment. It is also aware of both link and end-system limitations, and uses this information to set flow-control parameters [71].

TP++: TP++ is a transport protocol designed for multimedia applications that operates across high-speed networks. It makes provisions for multiple traffic classes and network scenarios, by being modular and composable, and is more flexible and complex than ISO-TP. TP++ was designed with large bandwidth-×-delay product paths in mind, and uses messages that are easily converted to different packet sizes to support heterogeneity. It also assumes that congestion control is provided by the network but provides mechanisms for end-to-end error control [72].

There are three major classes of applications that TP++ is designed for. *Constrained latency services* are those that are necessary for human interaction such as voice or video, in which the data sent becomes worthless if it does not arrive within a certain amount of time. *Transactions* occur mainly in distributed systems and are bursty in nature, but contain small to moderate amounts of data. The third class is *bulk data transfer* which carries large amounts of data between hosts. This generally requires reliable transfer but has looser latency constraints than the interactive data class.

TP++ connections are only one-way; bi-directional communication is accomplished using two one-way connections. TP++ uses not only ARQ, but also FEC for error control. Each 2^{16} byte (64 KB) *transport protocol data unit* (TPDU) has an associated 64-bit field

for detecting bit errors. The value of this field is computed at the sender and receiver using the WSC-2 [73] error detection code and the TPDU is assumed to be correct if the two values match. This is sufficient to detect all errors that change less than four bits. This relatively low number is based on the assumption that the underlying network links will be fiber-optic and have very low error rates. ARQ is used to recover losses due to congestion for the application classes that require reliable communications. The ACK messages used for ARQ include the last *connection sequence number* (CSN, a 32-bit byte counter) of all data that has been received contiguously since the beginning of the connection, as well as any CSN ranges received since then. This allows the transmitter to perform loss estimation and retransmit the necessary data, without retransmitting any correctly received data (selective repeat ARQ). It also allows the transmitter to maintain loss statistics and reduce the TPDU size in response to changing channel conditions, thereby creating a cross-layer control loop. When providing *constrained latency service*, TP++ can use FEC alone for error control. The FEC scheme used is only designed to correct packet drops, not bit errors, based on the assumption that congestion is the primary source of data loss. Each TPDU is segmented into *forward error correction blocks* FEBC's. Parity is then calculated by striping across the FEBC's. Missing FEBC's at the receiver can then be reconstructed using this erasure code, up to the number of additional FEBC's sent.

Connection management in TP++ is timer-based, which avoids the connection setup overhead of a handshake-based scheme. Connection setup is instantaneous when data is received at the destination, and shutdown occurs implicitly when the connection has been idle for a certain period of time.

RED: Random early detection is an implicit congestion notification mechanism. It is intended to avoid the network underutilization that occurs when a buffer overflow occurs resulting in a burst of packets being dropped from multiple TCP flow, causing them all to

back off. By randomly dropping a few packets when the buffer size reaches a threshold, several TCP flows are triggered to back off thus alleviating the congestion [74].

TCP-Vegas: TCP-Vegas is a sender-side modification of the earlier TCP-Reno implementations. It does not involve a change in the TCP specification, only in the implementation choices. TCP-Vegas uses a more accurate clock, and therefore is able to estimate the RTT more exactly than previous versions of TCP. This means it is able to respond to timeout events much sooner. TCP-Vegas is probably best known for its congestion avoidance mechanism, which operates by calculating an expected throughput equal to the congestion window size, divided by the minimum observed RTT. As long as the actual throughput is close to the expected throughput the congestion window is increased, however if the actual throughput drops significantly below the expected throughput it indicates that the congestion window is too large and congestion is being experienced on the path (even if no packets have been lost yet) so the window size is reduced. Lastly, TCP-Vegas modifies the slow-start mechanism to be less aggressive. It doubles the windows size only every other RTT, and compares the actual to expected throughput in between. When the actual starts dropping below the expected it changes to an additive increase instead of exponential increase in window size [41].

T/TCP: TCP extensions for transactions is designed for efficient handling of request-response oriented applications. The primary modification made to standard TCP, is that after the first connection between a given pair of hosts, subsequent connections may be made without completing a three-way handshake. This makes sessions consisting of only a few packets much more efficient. To enable this, the hosts must cache the parameters negotiated during their first connection after it is finished, allowing them to choose a new connection count number and resume at previous window values without renegotiating [75].

I-TCP: Indirect TCP is designed to allow mobile hosts to communicate with peers on fixed networks via mobility support routers, or gateways. The gateway maintains a standard TCP connection to the fixed host, on behalf of the mobile host, and the mobile host communicates with the gateway via I-TCP. I-TCP then applies a number of enhancements to the wireless link, including distinguishing between channel errors and congestion. It also handles handoffs from one gateway to another [76].

Snoop-TCP: Snoop-TCP is designed to enhance TCP performance in multihop wireless networks. It does not modify TCP itself, but instead changes the base station behavior. When running snoop, each base station caches unacknowledged TCP segments, and intercepts duplicate ACKs, performing the retransmissions locally [77].

TCP-NewReno: TCP-NewReno is an enhancement to the *fast recovery* phase found in TCP-Reno. Instead of waiting after resending the lost packet, it continues transmitting a new segment for every duplicate ACK received to keep the window full. If an ACK is received making only partial progress through the outstanding window NewReno assumes that is indicating another lost packet and retransmits that on next. This allows it to maintain much higher throughput after losses occur compared to TCP-Reno [40].

RTP: The real-time transport protocol is designed for delivering audio and video over IP. It does not handle error control or congestion, being UDP-like in behavior. It simply provides a standardized header for timestamps, sequence number, and a number of flags that ar commonly useful to multimedia streaming applications. It is typically implemented as a UDP shim and used in conjunction with a control protocol such as H.323, SIP, or RTCP [78, 79].

RMTP: The reliable multicast transport protocol provides in-order reliable deliver of data form a single source to multiple destinations. An end-to-end selective repeat algorithm is used, however the acknowledgements are aggregated on the return path through

the multicast tree, so that the source does not get overwhelmed with ACKs from all the destinations [80].

SCPS-TP: The space communication protocol standards transport protocol is a set of extensions and modifications to TCP to improve operation in the space environment. It both adds mechanisms to deal with specific environmentally-induced problems, and modifies existing mechanisms to reduce undesirable behaviors. The use of the SCPS-TP options is negotiated at the time of connection establishment, which allows the SCPS-TP agent to emulate TCP when communicating with a non-SCPS peer [81].

In SCPS-TP the default loss assumption is a user-selectable parameter on a per-path basis, so it will not assume congestion on links where congestion is unlikely. It also allows for signaling of congestion, corruption, and link outage both from the destination host and intermediate routers to explicitly determine the source of packet loss. SCPS-TP implements the TCP Vegas [82] slow start algorithm and congestion control based on RTT estimates. Additionally SCPS-TP queries the user for the path bandwidth-×-delay product and enters congestions avoidance once the congestion window size reaches this value, similar to the congestion avoidance algorithm described in [83]. This is beneficial for paths with high RTT. Explicit Congestion Notification (ECN) is done using source quench SCMP (SCPS specific version of ICMP) messages as in [84]. SCPS-TP also uses an open-loop token bucket rate control mechanism [85] for each space link to avoid congestion, with the available capacity shared in the global routing structure. For loss due to corruption, SCPS-TP relies on the ground-station at the receiving end of each space link to maintain a moving average of the ratio of corrupted frames received and to use explicit cross-layer messages to inform the SCPS-TP destinations when that ratio exceeds a threshold. The destinations are then responsible for continuously notifying their respective sources of the corruption, during which the sources will not reduce the congestion window or back-off the retransmission timer in response to packet loss. In the

case of a link outage, SCPS-TP assumes that the outage is bi-derectional, so the endpoints of the space link are responsible for notifying the SCPS-TP source and destination nodes on their side of the link. SCPS-TP then enters a persist state in which it periodically probes for link restoration at which point it can resume transmission where it left off without multiple timeouts or retransmissions or going through slow-start again.

To deal with the problem of highly asymmetric channels, SCPS-TP reduces the number of ACKs required by TCP [86] from every other segment to only a few per RTT. This requires other TCP mechanisms such as fast retransmit [87] to be disabled. To deal with constrained bandwidth in general, SCPS-TP employs header compression and Selective Negative Acknowledgment (SNACK) [88, 89]. The header compression is end-to-end, as opposed to the TCP/IP header compression specified in [90] that is done hop-by-hop. This is because hop-by-hop header compression requires a costly resynchronization process and looses all segments in flight every time a packet is lost or arrives out of order. The end-to-end compression achieve about 50% reduction in header size by summarizing information that does not change during the course of the transport session. It also avoids the problems incurred by changing connectivity because the compression takes place at the endpoints which remain constant. The SNACK option allows a single NAK [91] to to identify multiple holes in the the received data out-of-sequence queue. SCPS-TP also uses TCP Timestamps [92] to keep track of RTTs even with lossy channel conditions, and uses the TCP Windows Scaling option [92] so that the channel can be kept full even while recovering from losses.

M-TCP: Mobile TCP is designed for mobile cellular hosts communicating with peers on fixed networks. It makes use of a gateway that connects multiple cells to the fixed network to split the TCP connection. On the wired side, standard TCP is used, while M-TCP is used between the mobile host and the gateway. The gateway is responsible for assigning a fixed bandwidth to each mobile host within the cells it controls. The

gateway also preserves the end-to-end semantics of TCP, in that it does not acknowledge a packet until it receives an acknowledgement from the receiver. If the mobile device gets disconnected, the gateway advertises a receiver window of 0 to the fixed peer, putting it into persist mode. When the mobile host reconnects, the gateway advertises the normal size window to the sender, allowing the connection to resume with no back-off [93].

Mobile-TCP: Mobile-TCP is designed to be a low-complexity protocol suitable for mobile devices. It is capable of connecting to standard TCP peers via a single-hop wireless network and a mobile gateway that translates between the two protocols. A variant of header compression is used between the gateway and the mobile device to reduce the amount of data transmitted wirelessly. Error-correction in Mobile-TCP is asymmetric. From the gateway to the mobile device, go-back-n is used due to it's minimal processor and buffering requirements on the receiver side. When the mobile device is transmitting to the gateway, selective repeat is used to minimize the transmissions required. No congestion control or reordering capability is provided, due to the characteristics of the single-hop wireless evironment [94].

TCP-F: TCP Feedback is a cross-layer modification to notify the TCP sender of route failures in MANETs. This allows it to distinguish between losses due to route-failures, and losses due to congestion. When notified of a route disruption, the sender enters a snooze state where all timers are frozen. When the route is restored, the sender is notified again and resumes transmission without any back-off [95].

TCPSat: In 1999 the Internet Engineering Task Force (IETF) TCP Over Satellite Working Group (TCPSat) produced an informational RFC describing the issues affecting TCP performance of satellite links and identifying existing mechanisms that mitigate these effects [96]. These mechanisms were restricted to those mature enough to have reached RFC status, and which were backward compatible with TCP. In addition to the issues

and mechanisms already discussed, TCPSat recommends the use of path MTU discovery [97]. This allows TCP to determine the largest packet size that can be sent over the path without being subjected to fragmentation. Path MTU Discovery works by probing the path with progressively smaller packets that have the don't fragment (DF) bit set until it is able to reach the destination. The cost of doing this is significant delay (several RTT) before TCP can begin transmitting data, however by caching the path MTU for various links this cost can be amortized over multiple TCP connections. The assumption in doing path MTU discovery is that the effective BER is low enough that it is desirable to use the largest segment size possible to obtain maximum efficiency.

A related recommendation that is made is to use strong enough FEC that nearly all loss is due to congestion, since that is TCP's assumption when a packet is lost. TCPSat does not recommend doing loss discrimination and emphasizes the ned for TCP's congestion control behavior to avoid congestive collapse [62, 98]. The TCPSat recommendations only consider paths which originate and terminate on earth, passing through either a Geostationary Orbit (GEO), Medium Earth Orbit (MEO), or Low Earth Orbit (LEO) satellite, so the maximum RTTs being considered are only on the order of a second in contrast to the delays being considered in the SCPS-TP work.

The TCPSat group produced a second informational RFC outlining current research (circa 2000) that might improve TCP over Satellite links, but was not yet mature enough to be recommended [99].

STP: The satellite transport protocol incorporates many of the proposed TCP enhancements for satellite links including selective negative acknowledgments (SNACK), elimination of the retransmission timeout, periodic aggregated acknowledgements, and rate-based congestion control. Unlike SCPS-TP, STP does not attempt to differentiate between losses caused by congestion, and those caused by bit-error corruption [100].

Paced TCP: Paced TCP is flow control modification to TCP that spaces the packets allowed by the congestion window in time, instead of sending them all back-to-back during slow-start. The intent is to reduce the burstyness of traffic, to both improve fairness and overall bandwidth utilization [101].

2000–2002

SCTP: Stream control transmission protocol is designed to be a reliable protocol, with more sophisticated message-passing capabilities than TCP. It is intended for applications such as VoIP and ISDN over IP using multiple byte-streams (as opposed to TCP's single byte-stream). Using multiple byte-streams in parallel avoid the head-of-line blocking problem, and is able to achieve much better good put than a single stream over lossy links. SCTP can also make use of multi-homing by assigning multiple IP addresses to each end of an association, thus allowing for a hot-standby interface should the primary fail [102].

Freeze-TCP: Freeze-TCP is designed to improve TCP performance between mobile devices and hosts on fixed networks, without splitting the connection at a gateway, or requiring changes to the TCP code on the fixed node. When the mobile device is the receiver, it uses signal-strength information from the device's radio to predict a disconnection or handoff and advertises a zero winder size just before this happens. This forces the TCP sender in the zero-window-probing mode until it receives an ACK with a non-zero window advertisement. Since the TCP probes back off exponentially, the mobile receiver does not wait to receive a probe when it is reconnected, instead it sends 3 ACKs for the last packet received initiating fast-retransmit at the sender and resuming the connection [103].

TCP-BuS: TCP buffering capability and sequence information is designed to handle

route failures in MANETs. When the route fails, timer are extended, and packets in-flight are buffered. Any packets lost as a result of the route failure are retransmitted without reducing the congestion window [104].

PGM: Pragmatic general multicast is designed for applications such as multi-receiver file delivery where in-order reliable delivery is required. It uses negative acknowledgements sent to the source or designated repair nodes. As each negative acknowledgement is forwarded it is confirmed on a hop-by-hop basis [105].

ARC: Adaptive rate control is a hop-by-hop congestion-control mechanism for wireless sensor networks. Using an AIMD approach, each node continues to increase its sending rate as long as it continues to overhear its downstream node forwarding its packets. If it does not overhear its packets being forwarded, it backs off. It does this independently for both source and transit traffic to improve fairness [106].

ATCP: Ad hoc TCP is a cross-layer mechanism that manipulates TCP's state based on network-layer feedback. If an ICMP *destination unreachable* message is received, ATCP puts the sender into persist state where timers are frozen. It then probes the network until a new route is found. If an ECN message is received, ATCP invokes TCP congestion control immediately. If three duplicate asks are received, ATCP drops the third ACK, puts the sender into a persist state, and then retransmits the lost packet. When the next ACK is received, it returns TCP to the normal state [107].

TCP-RTO: Using a fixed RTO in TCP is designed to handle route failures in MANETs. Instead of using an exponential backoff, the RTO is fixed after the second timeout and remains fixed until the route is re-established and the packet is acknowledged [108].

TCP-Peach: TCP-Peach is designed for satellite channels with large bandwidth-×-delay products, as well as high BERs. It uses bandwidth estimates, and modifies both the slow start and fast recovery phases of TCP-Reno. In both cases it uses low-priority dummy

packets to probe the end-to-end path for additional available bandwidth. If the dummy packets are acknowledged with the same frequency as data packets, additional bandwidth is available, however if the dummy packets are lost with a higher frequency than data packets, congestion on the path is indicated. Dummy packets are marked with a low priority bit, requiring the routers on the path to drop packets preferentially based on this bit [109].

TCP-Peach+: TCP-Peach+ is an enhancement to the startup and recovery phases of TCP-Peach. It also replaces the dummy segments with NIL segments which carry unacknowledged data, while still being low priority. This allows for the recovery of corrupted packets without an explicit retransmission. In the startup phase, TCP-Peach+ sends 1 regular segment, and the receiver window – 1 NIL segments, and ramps up the congestion window to equal the total number of segments acknowledged using only 1 RTT. For loss recovery, TCP-Peach+ uses the SACK algorithm in addition to the methods described for TCP-Peach [110].

TCPW: TCP Westwood is also designed for lossy wireless environments, such as satellite links, but is a sender-side only modification to TCP-Reno. It uses the arrival-rate of acknowledgements to estimate the available bandwidth. It also smooths the estimate over time using a low-pass filter. Based on this estimate TCPW implicitly differentiates between corruption and congestion losses [111].

COPAS: Contention-based path selection is a network layer mechanism designed to increase TCP performance in MANETs. It uses two mechanisms, the first is choosing disjoint forward and reverse paths, so the data and ACK packets are not both causing contention on the same links. The second is monitoring the level of contention on each path, and selecting an alternate route if the contention on a given path is above a threshold [112].

Split-TCP: In unreliable networks such as MANETs, TCP tends to perform very poorly due to the high number of end-to-end retransmissions incurred and the congestion avoidance mechanism that is triggered by packet loss. Split-TCP divides long paths into seveal shorter onces, inserting proxies to interface between the segments. The proxies buffer, acknowledge, and retransmit packets and are able to improve performance by breaking up the end-to-end semantics of TCP [113].

XCP: The explicit control protocol is designed to handle large bandwidth-×-delay product environments. It uses an explicit router-based congestion-control feedback mechanism, thereby eliminating the need for the end-to-end layer to probe for available bandwidth [114].

PSFQ: Pump slowly fetch quickly is a reliable protocol designed for wireless sensor networks. Instead of the normal source-to-sink traffic model, it is designed for the case where the sink is transmitting to the sensor nodes, such as when retaking them or uploading new firmware images, and thus full reliability is needed. As the name implies, PSFQ uses a low sending rate, so that the normal network operation will not be impeded. If a packet loss is detected, the node uses a negative acknowledgment to rapidly request it from the upstream neighbor. In order to guarantee full reliability, an end-to end positive acknowledgement is used after all the message segments are received [115, 116].

WTCP: The wireless transmission control protocol is designed to improve throughput in wireless wide-area networks. It uses rate-based flow-control instead of ACK clocking to avoid injecting bursts of data into the network. It adjusts the congestion window based on the inter-packet arrival time at the receiver in steady state. If packets are lost, WTCP attempts to infer the cause. If the next received packet after the loss arrives at the expected time based on the packet arrival rate before the loss, the no congestion is inferred. However if it arrives later, then congestion is inferred and the sending rate is

halved. Instead of slow-start it uses the packet-pair approach to estimate the appropriate sending rate at startup [117].

ELFN: The explicit link failure notification technique is similar to TCP-F, in that a notification of route failure is sent to the sender, at which point it freezes all timers. In ELFN however, instead of waiting for a notification of rout restoration, the sender probes periodically and resumes transmitting if one of the probes is acknowledged [118].

TCP DOOR: TCP detection of out-of-order and response is designed to prevent unnecessary retransmissions when packets are delivered out-of-order in MANETs. This mechanism adds a few bytes to the TCP header, which are used to distinguish retransmitted packets from reordered packets at the receiver. If the receiver detects that packets were received out of order (likely triggering congestion avoidance at the sender) it notifies the sender of the out-of-order packets. The sender then immediately recovers to the state it was in before it entered congestion avoidance [119].

TCP-Probing: TCP-Probing is a sender-side modification to traditional TCP. When a loss occurs it pauses the data transfer and uses pairs of probing packets to determine if the path is congested. Based on this determination it either immediately recovers, or enters slow-start. During the probing cycle, which lasts at least 2 RTTs, no data is transferred [120].

2003–2004

HS-TCP: High-speed TCP is a modification to the congestion control mechanism of standard TCP to enable better performance in networks with large bandwidth-×-delay products. When the congestion window is small, it uses the AIMD algorithm from TCP-Reno to ensure fairness with other TCP variants, however when the congestion window is large it uses a lookup table to determine the increase and decrease values, meaning

that it increases more than one MSS per RTT and reduces by less than $\frac{1}{2}$ when a loss is encountered [121].

Scalable-TCP: Scalable-TCP is similar in intent to HS-TCP, that is faster adjustment with large window sizes. In the case of Scalable-TCP, it adds $\frac{1}{100}$ to the congestion window each RTT without congestion, and reduces by $\frac{1}{8}$ if congestion is experienced [122].

TCP Veno: TCP Veno is a sender-side modification to TCP Reno designed to improve performance in infrastructure wireless networks by discriminating between random losses and congestion losses and reacting appropriately. It uses the packet-pair method from TCP-Vegas to estimate queue sizes on the path. Because it relies on a measured RTT value, it is not expected to work well in MANETs or other environments with frequent route changes [123].

ILC-TCP: The interlayer collaboration protocol for TCP is designed to store the state of the link and IP layers in wireless networks, so that the TCP sender can request this information from ILC when a retransmission timer expires. If the lower lyres are not stable, then the ILC version of TCP assumes congestion was not the cause of loss and recovers immediately. This is only useful when the sender is directly connected to the lossy wireless channel, not when a lossy wireless link exists in the middle or receiver end of the path [124].

DelAck: The dynamic delayed ACK is a modification of the TCP delayed ACK option [86]. Instead of acknowledging only 2 packets at a time, they increase the delay as the sequence number increases, thus sending fewer and fewer ACKs as the session continues, and reducing congestion due to ACK traffic [125].

Link RED: Link random early detection monitors the average number of link-layer retransmissions occurring, and begins probabilistic dropping of packets according to the RED algorithm [74] above some threshold [126].

Neighborhood RED: Neighborhood random early detection is designed to improve the fairness of the RED algorithm in MANETs. Since congestion of the wireless channel is a function of all the node queues in the area, performing RED at each node independently may unfairly penalize some flows. In neighborhood RED, each node computes an average queue size calculation based on its own queue as well as its upstream and downstream neighbors, and then uses the RED algorithm to probabilistically drop packets [127].

Adaptive Pacing: Adaptive pacing is designed to improve spatial reuse in MANETs by increasing the backoff period after transmission, to allow the next hop to forward each packet without contention for the channel. It can be used in conjunction with link RED, which switches from the standard 802.11 backoff to adaptive pacing when the link retransmission threshold is reached [126].

NWCS: Non work-conserving scheduling is designed to improve fairness among TCP flows crossing heterogeneous networks made up of wired and wireless ad hoc networks. It does this by increasing the backoff period for queues outputting at a high rate [128].

DTN-BP: The delay tolerant networking bundling protocol is designed to improve performance when large and variable delays, intermittent connectivity, and high error rates exist in the network. It is a store-and-forward application-layer overlay that sends packages of data over a wide range of underlying network types and transport protocols, using a sequence of gateways that serve as nodes in the overlay [129, 130].

RMST: Reliable multi-segment transport is designed for wireless sensor networks. It uses hop-by-hop caching and repair to provide reliability and runs on top of directed diffusion. RMST also uses and end-to-end selective negative acknowledgment mechanism to guarantee full reliability [131].

CODA: Congestion detection and avoidance is designed for wireless sensor networks. It uses three mechanisms, congestion detection at the receiver, hop-by-hop backpressure,

and AIMD rate adjustment at the source. When congestion is detected via periodic channel monitoring, it begins broadcasting backpressure messages that are propagated upstream to the source, which adjusts it sending rate using AIMD or other local congestion policy [132].

ESRT: Event-to-sink reliable transport is designed to reduce congestion and power consumption in wireless sensor networks. The authors use the term reliability in their paper to refer to the sensor's reliability in detecting an event, based on its reporting frequency, which has resulted in some literature referring to ESRT as a reliable protocol, however this is not to be confused with packet reliability, since lost packets are not retransmitted. ESRT simply attempts to maintain a source reporting rate just high enough to detect events, without using excess power or inducing congestion. Nodes set a congestion bit on outgoing packets if their buffer overflows, and when the sink receives packets with the congestion bit set it adjusts the reporting rate via a high-power radio broadcast [133].

SRTP: The secure real-time transport protocol adds AES encryption to the original RTP, as well as using a hashing algorithm to authenticate each packet, protect its integrity, and prevent replay attacks. SRTP is controlled by SRTCP. [134]

P-XCP: Proportional XCP is a modification to XCP, which is designed to enable it to perform well in a high BER environment. To do this it modifies the sender to not back off when packet loss is experienced, adjusting the data sending rate based only on the explicit congestion notifications. It also increases link utilization by modifying the aggregate congestion feedback mechanism to account for rate limited connections [135].

TP-Planet: TP-Planet is designed for deep-space backbone links of the interplanetary network, and to outperform other TCP variants using the congestion-control algorithm used in TCP-Vegas and SCPS-TP. Instead it uses a rate-based AIMD congestion control

algorithm. Instead of TCP slow-start, TP-Planet uses *initial state*, which rapidly ramps up the sending rate to the slow-start threshold using only a fraction of an RTT, instead of the many RTTs that would normally be required. After the initial state algorithm is complete, the *steady state* commences, during which time the path is continuously probed by the receiver sending minimum-size (40 byte) segments with alternating high and low priority. By monitoring the receipt of these packets the receiver can determine if there is congestion on the path (more low-priority losses than high) or if there is only corruption-induced loss [136].

PETRA: The performance enhanced transport architecture uses the split-TCP policy to divide the end-to-end connection and use different transport protocols in each realm. It identifies two sublayers within the transport layer, the lower transport layer (LTL) that is responsible for error recovery within each realm, and the upper transport layer (UTL) that is responsible for end-to-end reliability [137].

GARUDA: GARUDA (named after a flying creature that appears in various Asian mythologies) is designed to handle reliable data delivery in wireless sensor networks. It constructs a two-level hierarchy consisting of core and non-core nodes. Core nodes elect themselves based on the hop count of the packet they received, if they have not heard from any other core node, to ensure that they are sufficiently distributed. As a message is propagating through the network, core nodes cache a full copy of the data being transmitted and use negative acknowledgments to fill any holes. Once the core nodes have received all packets in a message they broadcast a notification (an *A-map*) to the surrounding non-core nodes, which in turn use negative acknowledgments to retrieve any missing packets from the core nodes [138].

CFDP: The CCSDS File Delivery Protocol is another store-and-forward application-layer overlay, similar to DTN-BP, and uses TCP and UDP to transfer files from node to

node. CFDP offers both reliable and unreliable service options, with the reliable mode operating over UDP and depending primarily on several selectable negative acknowledgment algorithms at the application layer to effect the retransmission of lost data. The unreliable mode operates over TCP, leaving the reliability to be handled by the transport layer [139].

DTC: Distributed TCP caching is a TCP enhancement for wireless sensor networks that does not modify the sender or the receiver, only intermediate nodes. The goal is to reduce energy consumption, by reducing the number of end-to-end retransmissions. To this end, DTC caches 50% of the TCP segments at each intermediate node, and reads the SACK options of the TCP header in order to retransmit lost packets from its cache [140].

Fusion: Fusion is a multilayer approach to handling congestion in wireless sensor networks. It includes rate-limiting at the source, hop-by-hop flow control, and includes a prioritized MAC layer designed to allow sues at congested nodes to drain. Each node sets a congestion bit in outgoing packets, that is overheard by surrounding nodes, thus eliminating a need for explicit congestion notifications [141]. In a process referred to as backpressure, each upstream node from the one that is congested is allowed to transmit one additional packet, in which it in turn sets the congestion bit.

CCF: Congestion control and fairness is designed for wireless sensor networks, and is intended to provide better fairness than ESRT and CODA are able to in a congested environment. It divides the available bandwidth at each node across all upstream nodes, resulting in an exact hop-by-hop allowable rate calculation [142].

Trickle: Trickle is designed to handle congestion in wireless sensor networks using a hop-by-hop rate adjustment. The mechanism used id called *polite gossip* and suppresses transmissions by the local host if it has overheard more than some threshold of transmissions containing the same information by its neighbors [143].

ETEN: Explicit transport error notification is an approach to adding loss-discrimination to TCP for error-prone wireless channels. Channels with non-negligible packet corruption rates can degrade the performance of TCP significantly due to the assumption that congestion is the source of all packet losses. If the TCP source can distinguish between loss due to congestion, and loss due to corruption, it can respond appropriately in each case and significantly improve performance. Because Internet routers are required to drop corrupted packets immediately, explicit error notification requires alterations to intermediate hop behavior, not just the sender and receiver [144].

REFWA: Recursive, explicit, and fair window adjustment is a TCP modification designed for multihop satellite networks. This proposal is unique in that the TCP protocol remains unchanged, but the fields in the TCP headers are manipulated at intermediate nodes in order to improve efficiency and fairness in the network. To do this, REFWA matches the sum of the windows sizes of all TCP flows sharing a bottleneck link, and then adjusts the receivers advertised window field in each TCP ACK, in proportion to the estimated RTT for each flow [145–147].

mTCP: Multi-path TCP is designed to stripe a data flow across multiple redundant network paths in order to aggregate the bandwidth of those paths. It includes a shared-congestion detection algorithm to avoid taking a unfair share of a bottleneck link shared by two paths. The basic design consists of a single TCP-Reno with SACK flow on each path with the send and receive buffers being shared across the TCP flows. The mTCP sender maintains a scoreboard to track which path is used for each packet and detect failing paths. All acknowledgements use a single path in order to minimize ACK reordering [148].

TCPW+: TCP Westwood+ is a modification of TCP-Westwood, designed to smooth the bandwidth estimation component. It does this by measuring the bandwidth once

every RTT, instead of every time an acknowledgement is received [149].

TCP-Jersey: TCP-Jersey combines the TCP-Reno protocol with the bandwidth estimation of TCP-Westwood and router-based explicit congestion notification [150].

H-TCP: Hamilton TCP is designed for long fat networks. It modifies the AIMD algorithm by increasing more aggressively in proportion to the time elapsed since the last congestion event. The backoff is set proportionally to the inferred buffer size at the network bottleneck(s) based on the minimum and maximum observed RTT [151].

JTCP: Jitter-based TCP measures jitter at the receiver, relative to the segment timestamps, to determine the level of congestion on the path. When the jitter ratio is above a threshold, congestive loss is assumed and JTCP behaves like TCP-Reno, however if non-congestive loss is detected the window size is only reduced by a small factor [152].

TCP-ADA: TCP with adaptive delayed acknowledgement is similar in intent to DelAck, i.e. it seeks to reduce the number of asks required, in this case to reduce channel contention. Instead of delaying acknowledgments for a certain number of segments, TCP-ADA uses a timer-based method for deferring ACKs, where the timer is extended as long as packets arrive regularly, usually for an entire congestion window of data. This has some negative side-effects, including susceptibility to ACK loss [153].

2005–2006

RBC: Reliable bursty convergecast is designed for wireless sensor networks that experience large bursts of traffic following the detection of an event by multiple nodes. It uses randomized timers to reduce contention, and also allows for out-of-order acknowledgment of packets [154].

Siphon: Siphon is a multilayer approach to handling congestion in wireless sensor networks, which involve deploying a set of nodes within the sensor field that are able to act as virtual sinks and *siphon* traffic away from congested areas of the network and forward it to the sink via a long(er)-range wireless network on a secondary channel [155].

STCP: Sensor transmission control protocol is a generic reliable transport protocol for wireless sensor network. Generic in this case means that it is designed for much more opaque layer boundaries than many sensor network protocols, so it does not perform hop-by-hop functions, nor control application behavior. STCP is connection oriented, and uses negative acknowledgments for continuous data flows, and positive acknowledgments for event-driven flows. It also uses explicit congestion notification, with the network layer setting a congestion bit in the STCP header, which is read by the sink similar to DECbit. The sink then notifies the sender of congestion in an acknowledgment packet [156].

REFWA Plus: Recursive, explicit, and fair window adjustment plus adds a loss discrimination component to the TCP sender, based on the feedback provided by the original REFWA algorithm. When packet loss is detected, the sender checks the current indicated receiver window (set by REFWA) agains the previous value. If the new value is smaller the sender takes the loss as an indication of congestion and enters the fast-retransmit phase as usual, however if the loss is not accompanied by a reduced receiver window the sender simply retransmits the lost packet within the current window, assuming the loss was due to a channel error [157].

XSTP: Extended STP combines the TCP-probing mechanism with STP, in order to differentiate between losses caused by congestion, and those caused by bit-errors [158].

PORT: Price-oriented reliable transport attempts to minimize energy consumption while achieving the necessary reliability for an application. It uses the event-to-sink concept of reliability introduced in ESRT. PORT assigned a price to each node, based on the number

of transmissions required to *successfully* transfer a packet from that node to the sink. Low cost nodes are then preferred in routing, thereby reducing energy consumption [159].

TCP-Casablanca: TCP-Casablanca is based on TCP-NewReno and uses a statistical means of discriminating between random losses and congestion-induced losses. 1 in k packets is marked, and the routers are modified to drop marked packets first when congestion occurs. If the number of marked packets is disproportionately high, the receiver determines congestion to be the cause and standard NewReno behavior follows, however if the losses are randomly distributed the receiver sets the explicit loss notification flag on the ACK so that the sender will not back off [160].

TCP-DCR: Delayed congestion response TCP is designed to avoid backing off in case network reordering, not congestive loss, is the case of out-of-order segments at the receiver. The delay is configurable, and set to one RTT by default [161].

ATP: The ad hoc transport protocol is a non-TCP derived design for MANETs. It makes use of cross-layer information to respond to route failures. Congestion control is performed based on explicit feedback from the network, while reliability uses the selective acknowledgment algorithm, so the two are not interdependent. Each node updates the ATP header of forwarded packets with its queuing and transmission delay values, if they are higher than the existing value. The receiver is then able to average these values and determine if the network is becoming congested [162].

TCP-Westwood-BBE: TCP-Westwood with buffer and bandwidth estimation is designed to enhance TCP-Westwood's friendliness to existing protocols such as TCP-Reno. To do this, TCP-Westwood-BBE estimates the maximum expected RTT before the bottlneck link's buffer overflows. If a loss occures when the current RTT is close to the max, the loss is assumed to be due to congestion and the window size is reduced by $\frac{1}{2}$. If the current RTT is near the minimum observed RTT then the loss is assumed to be random

and the congestion window is not reduced. At in-between RTT values, the congestion window is reduced proportionately [163].

TCP-AReno: TCP-AReno is based on TCP-Westwood-BBE, tracking the minimum RTT and the RTT observed right before congestion loss events. It manages the congestion window in two parts. The base part follows TCP-Reno AIMD algorithm. The fast probing part is set close to the estimated bandwidth of the bottleneck link as long as the RTT stays close to its minimum value. When packet loss occurs, the multiplicative backoff is scaled based on the current RTT, as in TCP-Westwood-BBE [164].

DST: Delay sensitive transport is designed for wireless sensor networks and uses the concept of event-to-sink reliability introduced in ESRT. It introduces a delay bound, and tunes the reporting rate such that the sink will receive a sufficient number of packets to reliably indicate an event within that delay bound, and without introducing congestion [165].

PCCP: Priority-based congestion control protocol is designed for wireless sensor networks. It compares the packet-interarrival time with the service time of each packet on a hop-by-hop basis as an indication of congestion. If a node is congested it inserts this information into the header of forwarded data packets so that it will be overheard by upstream nodes, similar to the backpressure mechanism used by other protocols [166].

MRTP: The multiflow realtime transport protocol is designed to improve service quality for realtime multimedia applications. MRTP is controlled by MRTCP, and is designed to run on top of UDP or SCTP. Data is striped across the available paths and resequenced at the receiver. MRTP is connection oriented, and requires a 3-way handshake to establish a session. Paths are added and removed on the fly as necessary, and positive acknowledgments are used for control messages [167].

DTN-LTP: The Licklider transmission protocol is a point-to-point unidirectional protocol that deals with individual long delay links by freezing timers that would otherwise expire before an acknowledgement was received. It can transmit both reliable and unreliable data simultaneously, requiring acknowledgments for the reliable data only. It relies on a lower layer scheduler to tell it exactly when and how much to transmit, and when and how long delays will be. Because it is only designed for dedicated point-to-point links LTP does not handle congestion or routing issues [168–172].

2007–2008

Flush: Flush is a receiver-initiated reliable bulk transport protocol for wireless sensor networks, which uses end-to-end negative acknowledgements and an integrity check of the entire transfer for reliability. During connection setup it determines the number of hops in the path, and optimizes the sending rate to maximize spatial reuse along the path. It assumes that only one transfer occurs at a time so that inter-flow interference or congestion are not a factor [173].

RCRT: Rate-controlled reliable transport is designed for highly-utilized wireless sensor networks that are loss intolerant. It uses end-to-end negative acknowledgments, as well as cumulative positive acknowledgements piggybacked on other feedback packets to ensure full reliability. RCRT also employs centralized rate-control to avoid congestion, by assuming that the network is uncontested as long as lost packets can be retransmitted quickly. If that is not the case, the sensors are instructed to reduce their sending rate [174].

ART: Asymmetric and reliable transport is designed for wireless sensor networks, and exploits the fact that event-to-sink communications (often) do not have the same reliability requirements as sink-to-sensor transmissions. It uses end-to-end negative acknowledg-

ments to fill gaps in sink-to-sensor data, with a positive acknowledgment for the last packet in each message. ART uses the concept of event reliability from ESRT when dealing with event-to-sink transmissions, so positive acknowledgments are used, but only on a per-event basis, not on a per-packet basis [175].

DTSN: Distributed transport for sensor networks is designed to offer two reliability modes, depending on application requirements. It is connection oriented, but connection setup is implicit. In the fully-reliable mode DTSN uses window-based loss detection, with both negative and positive acknowledgments. In the case of a negative acknowledgement, each intermediate node searches its local cache and attempts to fill the holes identified in the NACK, updating the NACK packet accordingly before forwarding it. In the differentiated reliability mode, the application is allowed to denote *core* data (e.g. key frames in a video sequence) and only the core data is preferentially cached at the source. The remaining data is left to hop-by-hop reliability mechanisms or best-effort service. DTSN does not include a congestion-control mechanism [176].

PHTCCP: Prioritized heterogeneous traffic-oriented congestion control protocol is designed to handle congestion in wireless sensor networks supporting multiple application types. It uses the ratio of the packet service time, to the packet scheduling time at the MAC interface as the indicator of congestion at each node. Each node piggybacks its congestion information on forwarded nodes so that upstream nodes can overhear it and reduce their sending rate if needed. Priority scheduling is handled at the MAC layer, where high-priority packets use shorter backoff timers for contention and smaller interframe spacing [177].

CTCP: Collaborative transport control protocol is designed to handle reliability and congestion control in wireless sensor networks. It is connection oriented, using a handshake to set up the connection and establish the flow ID. CTCP uses hop-by-hop reliability,

but has two modes. In the first mode each node receives and acknowledgement from the next node. In the second mode, each node receives an acknowledgement from the next 2 nodes. Congestion is controlled by broadcasting a stop signal if a node nears buffer capacity, and a start signal when the buffer is nearly empty. Each node maintains a table of the flows transiting it, and the state of it downstream neighbors buffers [178].

RT^2: Real-time and reliable transport is designed to address the reliability needs of wireless sensor networks. They build on the idea of event reliability, as opposed to packet reliability, introduced in ESRT. In this case however, they also use selective acknowledgments to provide full reliability for control traffic [179].

PALER: Push aggressively with lazy error recovery is designed for code distribution or other one-to-many applications that require reliability. It is based on PSFQ, but reserves all error-correction until the end and handles it with a single negative acknowledgement. This is shown to require fewer total transmissions than PSFQ [180].

TCP-AW: TCP Adaptive Westwood combines the ideas of TCP-Westwood with TCP-AReno to achieve efficiency in high bandwidth-×-delay product networks, RTT fairness, and fairness with existing protocols. It detects congestion using RTT monitoring from Westwood, and reduces the congestion window to safe levels using the algorithm found in Adaptive-Reno [181].

TCP-Illinois: TCP-Illinois is designed for high-speed networks and is based on TCP-NewReno. It adapts the AIMD parameters based on inferred changes in queuing delay. When the RTT is close to the minimum observed, then queuing delay is assumed to be small and the congestion window is increased more aggressively, while the backoff due to loss is reduced to a small fraction. Conversely when the RTT increases, the queuing delay is assumed to be increasing and the congestion window is increased less aggressively, while the backoff due to loss is increased to a larger fraction [182].

MPLOT: Multi-path loss tolerant TCP is designed to make use of multiple redundant paths in highly lossy (up to 50% packet-error rate) environments. Similar to mTCP, congestion control is handled on a per-path basis using standard TCP mechanisms, however error control is spread across the paths using erasure coding to statistically reduce the number of retransmissions required. MPLOT continuously estimates the RTT, loss-rate, and available capacity of each path, and uses this to do latency-aware packet mapping [183].

2009–2011

ERTP: The energy-efficient and reliable transport protocol is designed for streaming data in wireless sensor networks where congestion is not a concern, but reliability is required. ERTP uses hop-by-hop retransmission, based on implicit acknowledgements, combined with the probability of packet lost on each link to determine the number of retransmissions that should be allowed on each hop to achieve an acceptable level of end-to-end reliability [184].

CRRT: Congestion aware and rate controlled reliable transport is designed for wireless sensor networks. It uses both hop-by-hop and end-to-end retransmissions for reliability, and includes a mechanism for centrally controlling the sending rate of each node to reduce congestion [185].

TRCCIT: Tunable reliability with congestion control for information transport is a wireless sensor network protocol that selects between ARQ and multipath options for tunable reliability. It uses both implicitly and explicit acknowledgments on a hop-by-hop basis, and timers to trigger transmissions if a packet is not acknowledged by the next hop. It does not use end-to-end acknowledgments. Nodes will also choose alternate paths if their next-hop neighbor is congested, resulting in partially disjoint paths [186].

TCP-Derwood: TCP-Derwood is designed to handle significant changes in available bandwidth and RTT which result in dramatically different bandwidth-×-delay products, and are caused by switching between terrestrial and satellite links. To detect a change in link type, the sender observes the sudden change in RTT, as well as other characteristics of the ACK timing. When switching to a satellite link, TCP-Derwood increases its congestion window to allow it to fill the new pipe, and when switching to a terrestrial link the congestion windows is decreased [187].

2.5 Diversity

Due to established layering paradigms, diversity is difficult to introduce at the transport layer in a meaningful way (unified designs for sensor networks as seen in the previous section being the exception), and instead it *typically* appears in multipath routing architectures where it operates on traffic aggregates, and is decoupled from specific application requirements or optimizations. That being said, there have been a number of these routing approaches which are relevant to our work on end-to-end diversity and we will look at those in this section. We also note that diversity in this context is significantly different than the wireless channel diversity used to enable network coding, and that although there is ongoing research on the effects of network coding on the end-to-end layer [188], that is a separate topic and considered outside the scope of this dissertation. We are interested in diversity due to its potential to improve network survivability.

2.5.1 Network Survivability

The study of network *survivability* is an extension of the study of *fault-tolerance*, which is the ability of a system to tolerate faults such that service failures do not result. Fault tolerance generally covers random single or at most a few faults, and is thus a subset

of survivability [6]. The current level of reliance on the Internet in modern nations led to the understanding that fault-tolerant designs were not sufficient and that *diversity* in multiple forms is needed to prevent multiple parts of the infrastructure from sharing fate and thereby protect against correlated failures.

Survivability is the capability of a system to fulfill its mission, in a timely manner, in the presence of threats such as attacks or large-scale disasters. This definition captures the aspect of correlated failures due to an attack by an intelligent adversary [15, 16], as well as failures of large parts of the network infrastructure [189, 190]. Note that disasters can be natural (e.g. hurricane, coronal-mass ejection, earthquake) or human caused (e.g. blackout, electro-magnetic pulse weapon).

Based on this definition, *survivability* may encompass a broad spectrum of failure scenarios, however the aspect about which we are concerned in this work is the ability of a topology to remain connected (the acceptable service) [191, 192] while undergoing multiple simultaneous node and link failures (due to external challenges) [7, 193].

2.5.2 Graph Theoretic Diversity Approaches

The problem of finding paths through a network has been well studied in the context of graph theory [194] as well as fiber network planning. The existing algorithms are based on different characteristics of these paths such as shortest paths, diverse, and disjoint paths [14], and optical restorability [195]. Several algorithms exist to find the shortest path or k-shortest paths that include the earliest shortest path algorithms by Ford [196], Moore [197], Dijkstra [198], and Floyd [199], along with several modifications that address negative cycles and improve on or in some cases trade time and space complexities [200]. Following the shortest path between a pair of nodes, several algorithms were proposed to

find the k-shortest paths, which involve simple techniques such as manipulation of edge weights to highly optimized algorithms [201].

Furthermore, the concept of diverse paths has been investigated to find a pair of diverse paths, k-diverse paths, and k-shortest diverse paths. The existing literature covers techniques based on shortest path algorithm with the incremental removal of used edges to graph transformations [202, 203]. Bhandari presents efficient algorithms to compute edge-disjoint and vertex-disjoint paths [200]. However, these algorithms are based on finding completely diverse paths. Bhandari also discusses an algorithm that finds the maximally diverse paths between a pair of nodes using a modified Dijkstra's algorithm.

In this work, we use a method that draws heavily on Bhandari's maximally-diverse paths. However, we consider *both edge and node* disjoint diversity. We further expand the diversity measure by applying it to a set of nodes and to a full graph. We apply our algorithm to a network scenario in which the objective is to find k paths such that the diversity of individual node pairs as well as the overall diversity of the network exceeds a minimum threshold. Work has been done to characterize ISP networks in terms of the redundancy present in their physical layer graphs [204]. Our work is consistent with these efforts when applied in the same context.

2.5.3 Multipath Routing

Multipath at the network layer does not have the same end-to-end semantics as it does at the transport layer, however there have been a number of proposed multipath routing approaches that may have counterpart mechanisms at the transport layer.

Path Splicing [5] uses multiple destination-rooted routing trees to provide multiple alternative paths that may be switched between at any intermediate node. In this approach, the source node is given control over selecting a path index at each intermediate hop,

however no information about the paths is passed to the source so there is no basis on which to chose a *particular* set of paths. The benefit is that if the source detects packet losses and suspects a bad link, it can randomly choose a different set of path indices much faster than routing can reconverge, however it has no assurance that the new path chosen will map to a different set of physical links. A similar approach is Routing Deflections in that the source node is given some control without detailed information [205].

Rope ladder routing (RLR) [206] combines the features of link, node, and path protection, wherein the primary and backup paths are connected at many intermediate points, not only at the source and destination. It uses GPS locations of the nodes to construct the rope-ladder paths, with the goal of quick recovery in the case of an outage on the primary path, and minimal packet loss.

An alternative, which is sometimes not even thought of as a multipath solution, is to have pre-computed back-up routes in case of a link failure [4, 207]. While this can be faster than a full reconvergence, it still takes time for the nodes at the location of the failure to detect it and begin using the alternate routes.

Much of the multipath routing research, such as DETOUR [208], has been focused on the ad-hoc wireless environment in which channel conditions and resource constraints are the primary concern.

2.5.4 Multipath Applications

There are a number of proposed application scenarios (e.g. video streaming [209, 210]) that recognize the benefits of scheduling packets across multiple disjoint paths for performance gain or to mitigate the effects of bursty channel errors. Our goal with respect to these scenarios is to expose a well-defined set of control parameters (knobs) that allow

the application to express its service level requirements without placing the full burden of discovering the optimal set of paths on the application itself.

2.6 Summary

Based on the literature survey, we can conclude that there are many approaches to learn from, in both the transport and multipath routing domains. The key difference between our approach to diversity and existing multipath routing research is that we explicitly look at it as an end-to-end problem, to be solved intelligently using cross layer information available at the endpoints, in the context of application requirements. Particular routing protocols are generally restricted to enabling multipath within an administrative domain, not controlling the end-to-end path. Our transport protocols draw on mechanisms originally presented in a number of the protocols here, but combining them into a unified protocol that is modular and adaptable to multiple network environments.

The following chapters will first present our approach to enabling diversity, followed by the design and analysis of our ResTP protocol, which uses diversity as well as a number of other mechanisms to improve resilience at the end-to-end layer.

Page left intentionally blank.

Chapter 3

Path Diversification

In this chapter we present *Path Diversification*, a new mechanism that can be used to select multiple paths between a given ingress and egress node pair using a quantified diversity measure to achieve maximum flow reliability. Diversity is one of the critical mechanisms identified in Chapter 1 for improving network resilience. Of the many forms diversity can take, we are looking at path diversity, by which we imply diversity of transit nodes and links. In order for path diversity to have a substantial impact on network performability, it must be available between a large percentage of node pairs, so we devote a significant portion of this chapter to the evaluation of the diversity available in network graphs.

The path diversification mechanism is targeted at the end-to-end layer, but can be applied at any level for which a path discovery service is available, e.g. intra-realm routing or inter-realm routing. Path diversification also takes into account higher level requirements for low-latency or maximal reliability in selecting appropriate paths. Using this mechanism will allow future internetworking architectures to exploit naturally rich physical topologies to a far greater extent than is possible with shortest-path routing or equal-cost load balancing. We describe the *path diversity* metric and its application at various aggregation levels, and apply the path diversification process to 13 real-world network graphs

as well as 4 synthetic topologies to asses the gain in flow reliability. Based on the analysis of flow reliability across a range of networks, we then extend our path diversity metric to create a composite compensated total graph diversity metric that is representative of a particular topology's survivability with respect to distributed simultaneous link and node failures. We tune the accuracy of this metric having simulated the performance of each topology under a range of failure severities, and present the results. The topologies used are from national-scale backbone networks, with a variety of characteristics, which we characterize using standard graph-theoretic metrics. The end result is a *compensated total graph diversity* metric that accurately predicts the survivability of a given network topology.

The work presented in this chapter has resulted in several publications. We proposed the original path diversity metric and path diversification algorithm in [211] and further evaluated it, along with the EPD and TGD metrics in [212]. More recently we proposed and evaluated the cTGD metric in [213], and we are preparing an extended version addressing all these elements as well as providing additional evaluation in [214]. The remaining sections of the chapter are organized as follows: In Section 3.1 we introduce path diversification and explain why it is an essential component for resilience in future network architectures. Section 3.2 presents the path diversification mechanism itself. Section 3.3 explains our evaluation methodology and presents our findings. Section 3.4 compares the survivability of our 17 topologies. Section 3.5 ranks these topologies and presents a new composite metric for predicting network survivability, and Section 3.6 concludes.

3.1 Motivation for Multipath

Many of today's networked devices have access to multiple partial or complete physical layer paths between endpoints, but are unable to benefit from them due to design decisions in the current Internet protocol stack that assume unipath routing. Depending on the application, the benefits of using multiple paths can be in the form of improved performance, increased dependability, or both. In this work we focus on the dependability aspects, recognizing that additional mechanisms are needed to optimize performance gains across multiple paths in real time (e.g. [215]), several of which we examine in Chapter 4.

The goal of path diversification is to provide a unified interface to a service that is as reliable as the underlying physical graph, instead of being limited by the reliability of a particular path as in the current model. This could alleviate much of the programming overhead that currently exists at the application level, providing a level of service not possible with the limited information currently provided to the end systems. At the same time, doing so requires that the end-to-end service be informed of application requirements in terms of throughput and upper delay bounds, given that there is an inherent tradeoff between maximizing path diversity and minimizing path stretch [5].

In the upcoming sections we present a formal definition of the *Path Diversity* metric, and its aggregate properties when applied to node pairs and complete network graphs. Based on this notion of diversity we then present *Path Diversification*, which is a heuristic algorithm designed to select the most advantageous subset of *available* diverse paths between two nodes in a network under particular application constraints. It yields several derived metrics reflecting some of the characteristics of the selected paths, as well as the network as a whole. We then explore how path diversification improves reliability by comparing it both to the conventional unipath approach and the real connectivity of the underlying

topology, which forms the upper performance bound in our case. We do not evaluate different path discovery mechanisms, but assume the availability of a path database that corresponds to the physical topology as has been proposed in the context of the Post-Modern Internetwork Architecture [24]. We then quantify the *survivability* of network topologies so that new or modified topologies may easily be compared quantitatively.

3.1.1 Terminology

While we try to maintain consistency with all widely accepted definitions, there are a few key terms defined here:

- Node pair: Any two nodes at the same hierarchical level of a particular network topology, e.g. two core nodes or two subscriber nodes.

- Path: Any complete set of nodes *and* links that form a loop-free connection between a node-pair.

- Path stretch: The ratio of the number of hops on a given path, divided by the number of hops on the shortest path.

- Flow: A data association between a node-pair which may be distributed over one or more paths.

- Application: The higher-level entity that specifies the service requirements of a particular flow. This may refer to a traditional software application, or an alternative motivating factor, such as an SLA (service level agreement) in the context of an ISP network.

3.1.2 Design Goals

In order to achieve resilience, an end-to-end flow should be able to exploit diversity to the degree that it is present in the physical network graph. Current unipath mechanisms that rely on shortest-hop single-path routing cannot accomplish this. A resilient multipath mechanism should have the following design goals in providing service to higher layers:

- **High flow reliability:** Once established, a flow should remain stable as long as the underlying physical network is not partitioned.
- **End-system control:** The end systems or application should have some control over the paths selected.
- **Optimal paths:** The paths chosen should be the best available given the application's service requirements.
- **Minimal impact:** There should not be a negative impact on the network as a whole.

Section 3.2.4 formally describes the algorithm used in path diversification to meet these goals.

3.2 Path Diversification Overview

The primary objective of path diversification is to select multiple paths that are as diverse as possible, while limiting the path stretch if necessary. Instantiating path diversification at any level will require the following four functions:

1. Path database indexed by source-destination pairs and including the unique identifiers for each node and link traversed. This database can be an exhaustive compilation or filtered based on administrative policy.

2. Quantified path diversity using the path diversity metric explained in the following section.

3. Path selection based on higher-level specifications to evaluate the tradeoff between path diversity and path stretch.

4. Packet forwarding based on the source routes distilled from the selected paths.

There are many possible implementations of the path discovery database and packet forwarding mechanisms, and path diversification is agnostic to this implementations, as long as the two are decoupled to allow greater flexibility in exploiting the inherent diversity of heterogeneous internetworks [24]. The following sections discuss the measurement of path diversity and the selection of diverse paths.

3.2.1 Path Diversity

Since the primary motivation for implementing the path diversification mechanism is to increase resilience, paths should be chosen such that they will not experience correlated failures. To this end, we define a measure of diversity (originally introduced in [211] and refined in [212]) that quantifies the degree to which alternate paths share the same nodes and links. Note that in the WAN context in which we are concerned with events and connections on a large geographic scale, a node may be thought of as representing an entire PoP, and a link as the physical bundle of fibers buried in a given right-of-way. This distinction between WAN and LAN component identifiers affects only the population of the path database, not the usage of the diversity metric.

Path: Given a (source s, destination d) node pair, a path P between them is a vector containing all links L and all intermediate nodes N traversed by that path

$$P = L \cup N \tag{3.1}$$

and the length of this path $|P|$ is the combined total number of elements in L and N.

Path diversity: Let the shortest path between a given (s, d) pair be P_0. Then, for any other path P_k between the same source and destination, we define the diversity function $D(x)$ with respect to P_0 as:

$$D(P_k) = 1 - \frac{|P_k \cap P_0|}{|P_0|} \tag{3.2}$$

The path diversity has a value of 1 if P_k and P_0 are completely disjoint and a value of 0 if P_k and P_0 are identical. For two arbitrary paths P_a and P_b the path diversity is given as:

$$D(P_b, P_a) = 1 - \frac{|P_b \cap P_a|}{|P_a|} \tag{3.3}$$

where $|P_a| \leq |P_b|$.

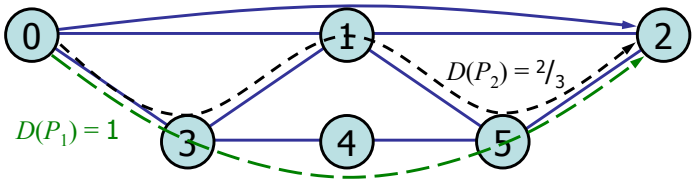

Figure 3.1: Shortest path P_0 and alternatives P_1 and P_2

It has been claimed [5] that measuring diversity (referred to as novelty) with respect to *either* nodes *or* links is sufficient, however we assert that this is not the case. Figure 3.1 shows the shortest path, P_0, along with the alternate paths P_1 and P_2 both of which have a (link) novelty of 1. However, given a failure on node 1, both P_0 and P_2 will fail. In our approach, $D(P_2) = \frac{2}{3}$, which reflects this vulnerability. P_1 on the other hand has

both a novelty of 1 and a diversity of 1, and does not share any common point of failure with P_0. Similarly, the wavelengths or fibers from multiple nodes may in fact be spliced into a single physical link such as was the case in the Baltimore Tunnel Fire [216, 217], resulting in a single point of failure, thus illustrating the need for including both nodes and links into the diversity measure.

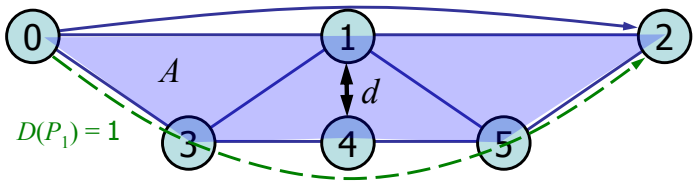

Figure 3.2: Geographic diversity: distance d and area A

3.2.2 Geographic Path Diversity

Where geo-location tags are made available through cross layering methods, we believe it is important to measure diversity in terms of physical distances, not only node and link disjointness. The previous EPD and TGD measures consider the sharing of components, but do not capture the geographic characteristics necessary for area-based challenges such as large-scale disasters or to prevent the geographic fate sharing of distinct links in the same conduit as in the Baltimore tunnel fire. Therefore we are augmenting the diversity measures with a minimum distance between any pair of nodes along alternate paths, and as the area inside a polygon or set of polygons, the borders of which are defined by a pair of alternate paths, as shown in Figure 3.2. Thus, it should be possible to specify diverse paths among a set of candidates with a given degree of sharing and distance metric EPD(d) constrained by stretch, and measure the geographic area between the paths EPD(A) as well as to measure the diversity inherent in a graph across all paths TGD(d, A). To further that end we propose the following definition of *geographic path diversity*

Geographic path diversity Given previously defined definitions of P_a and P_b

$$D_{\mathrm{g}}(P_b, P_a) = \alpha d_{\min}^2 + \beta A \tag{3.4}$$

where $d_{\min}$ is the minimum distance between any member of the vector P_a and any member of the vector P_b, and A is the area of a polygon whose borders are formed by paths P_a and P_b as shown in Figure 3.2. α and β are weighting factors in the range $[0, 1]$, chosen based on the application and whether the minimum distance or area are considered to be of greater concern.

3.2.3 Effective Path Diversity

Effective path diversity (EPD) is an aggregation of path diversities for a selected set of paths between a given node-pair (s, d). To calculate EPD we use the exponential function

$$\mathrm{EPD} = 1 - e^{-\lambda k_{sd}} \tag{3.5}$$

where k_{sd} is a measure of the added diversity defined as

$$k_{sd} = \sum_{i=1}^{k} D_{\min}(P_i) \tag{3.6}$$

where $D_{\min}(P_i)$ is the minimum diversity of path i when evaluated against all previously selected paths for that pair of nodes. λ is an experimentally determined constant that scales the impact of k_{sd} based on the utility of this added diversity. A high value of λ (> 1) indicates lower marginal utility for additional paths, while a low value of λ indicates a higher marginal utility for additional paths. Using EPD allows us both to bound the diversity measurement on the range [0,1) (an EPD of 1 would indicate an

infinite diversity) and also reflect the decreasing marginal utility provided by additional paths in real networks. This property is based on the aggregate diversity of the paths connecting the two nodes.

3.2.4 General Path Selection Algorithm

Given that the number of possible paths existing between a common (source, destination) pair is z:

Step 1. *Let A be the set of available paths between a given (source, destination) pair, in decreasing order by diversity value, where $|A| = z$*

Step 2. *Let n be the number of diverse paths required by the transport layer.*

Step 3. *Let B be the smallest subset of highly diverse paths, where $|B| = k$ and $k \geq n$.*

$$B = \{i \in A : D(P_i) > D(P_j), \forall j \in A\} \tag{3.7}$$

If $k = n$, B is the set of exactly n diverse paths required by the transport layer and the algorithm is finished, otherwise we continue with steps 4 through 8.

Step 4. *Let D_{min} be the minimum diversity amongst all paths in set B.*

$$D_{min} = \min[D(P_i), \forall i \in B] \tag{3.8}$$

Step 5. *Select a set C out of B which contains all the paths with a diversity greater-than D_{min}, where $|C| = m$*

$$C = \{i \in B : D(P_i) > D_{min}\} \tag{3.9}$$

Step 6. *Let D be the remaining paths in B after removing C, where $|D| = k - m$.*

$$D = B - C \tag{3.10}$$

Step 7. *Select set E, to be the shortest length paths from D, where $|E| = n - m$*

$$E = \{i \in D : |P_i| \leq |P_j|, \forall j \in D\} \tag{3.11}$$

This step allows us to choose shorter paths when path diversities are equivalent.

Step 8. *The final set S of n diverse paths is*

$$S = C \cup E \tag{3.12}$$

This algorithm yields the required number of paths with the constraint that they will include the shortest path and the maximally diverse paths with the least stretch.

3.2.5 Measuring Graph Diversity

The total graph diversity (TGD) is simply the average of the EPD values of all node pairs within that graph. This allows us to quantify the diversity that can be achieved for a particular topology, not just for a particular flow. For example a star or tree topology will always have a TGD of 0, while a ring topology will have a TGD of 0.6 given a λ of 1.

Here we note that for diversity to make sense in the graph context it should be computed considering only path components (nodes and links) at the level of network hierarchy for which the diversity value is desired. For example, in computing the diversity of a service provider's backbone, only core nodes should be considered, otherwise the comparatively vast number of subscriber nodes (typically stubs) will artificially reduce the calculated diversity. We also note here that the diversity measure is designed such that it does not penalize longer paths in favor of shorter paths, meaning that graph diameter and average path lengths are independent metrics that should be considered in addition to the diversity metric. This is discussed in further detail in Section 3.5.2.

3.2.6 Terminating Conditions

In this section we use three different modes for choosing a set of diverse paths for a given node pair: number of paths, diversity threshold, and stretch limit. The objective in the first mode is to find k maximally diverse paths. We first find the shortest fully disjoint paths, and if additional paths are required we continue finding paths that add maximum diversity as calculated using equation 3.6. The second mode selects as many maximally

diverse paths as are required to achieve the requested EPD. Finally, the third mode selects all maximally diverse paths with stretch less than the stretch limit. In all modes, the set of maximally diverse paths are found using the Floyd-Warshall algorithm with modified edge weights [200]. In this algorithm, only those paths are used that increase the EPD for the node pair in question. Recall that only paths with one or more disjoint elements (links,nodes) will result in non-zero $D_{\min}$ and consequently increase EPD.

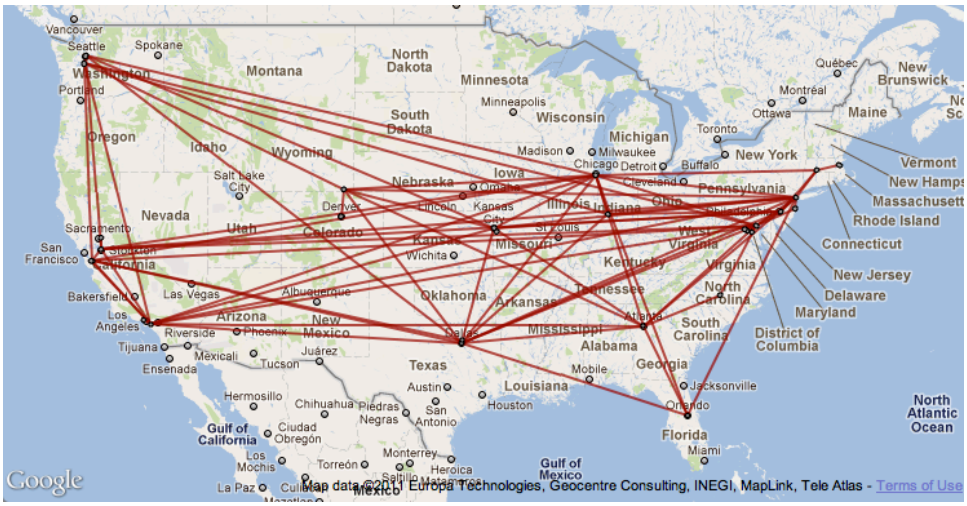

Figure 3.3: Sprint logical topology

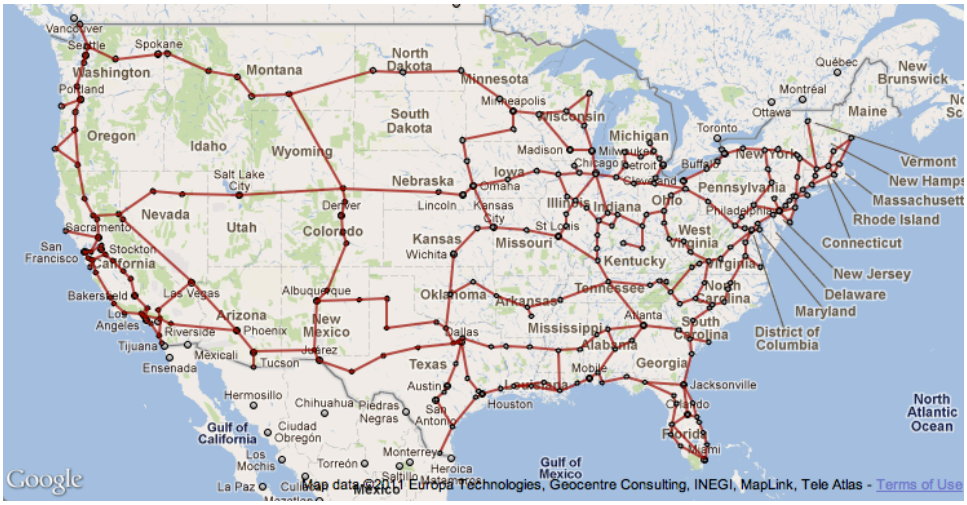

Figure 3.4: Sprint physical topology

Figure 3.5: VSNL logical topology

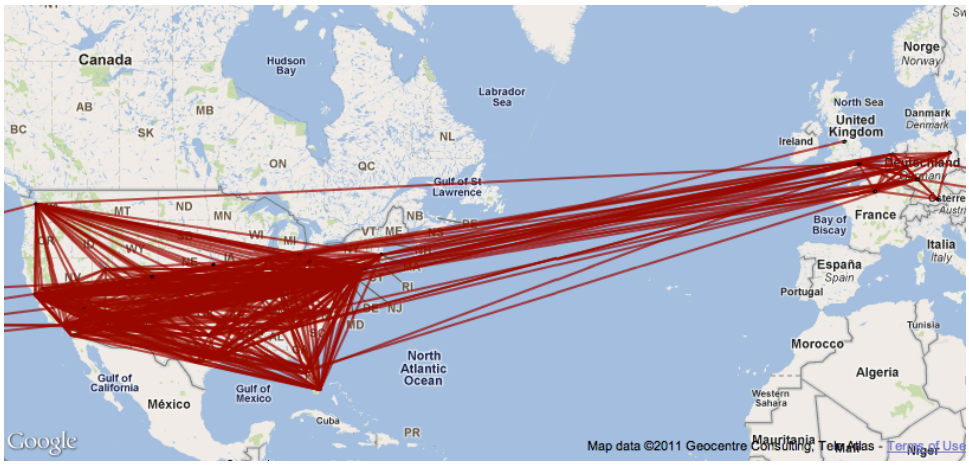

Figure 3.6: Level-3 logical topology

Table 3.1: Network Characteristics

Network	Nodes	Links	Avg. Node Degree	TGD $k=4$	Cluster. Coeff.	Diam.	Radius	Hopcount	Closeness	Node Between.	Link Between.
Full-Mesh	20	190	**19.00**	**0.9502**	**1**	**1**	**1**	**1**	**1**	**0**	**1**
Grid	25	40	3.20	0.8964	0	8	4	3.3333	0.3067	**110**	54
Ring	25	25	2.00	0.6321	0	12	12	6.5	0.1538	132	78
Star	25	24	1.92	0.000	0	**2**	**1**	1.92	0.5302	552	24
AboveNet	22	80	**7.27**	0.8559	**0.6514**	**3**	**2**	**1.7229**	**0.5947**	196	**21**
AT&T	108	141	2.61	0.5881	0.3274	6	3	3.3790	0.3030	4160	943
AT&T Phys.	361	466	2.58	**0.9014**	0.0550	37	19	13.57	0.0763	4527	1893
EBONE	28	66	4.71	0.8635	0.3124	4	3	2.2804	0.4507	132	42
Exodus	22	51	4.64	0.8843	0.3307	4	**2**	2.0563	0.4978	132	22
GÉANT Phys.	34	51	3.00	0.7623	0.2898	9	5	3.4652	0.3007	556	131
Level 3	53	456	**17.20**	**0.9154**	**0.7333**	4	**2**	**1.7721**	**0.5845**	664	84
Sprint	44	106	4.82	0.8120	0.3963	5	3	2.6882	0.3853	602	129
Sprint Phys.	263	311	2.37	0.8821	0.0340	37	19	14.78	0.0700	3609	1637
Telstra	58	60	2.07	0.1295	0.2411	6	3	3.3025	0.3095	2136	806
Tiscali	51	129	5.06	0.7785	0.5068	5	3	2.4298	0.4236	656	96
Verio	122	310	5.08	0.8104	0.3509	8	4	3.1026	0.3335	3736	480
VSNL	7	7	2.00	0.2001	0.4167	4	**2**	2.0952	0.4982	**18**	**12**

3.3 Evaluation

In this section we evaluate path diversification based on its ability to reflect the connectivity of the underlying graph, and the cost incurred in doing so in terms of path stretch. To evaluate path diversification based on realistic topologies we selected 13 service provider backbone network topologies (3 physical ant the rest logical), and 4 synthetic topologies. We selected a range of connectivity as shown in Figures 3.3, 3.4, 3.5, and 3.6 in order to evaluate path diversification on topologies with a broad set of property values. Maps of the remaining service-provider topologies may be found in Appendix C. Comparing Figures 3.3 and 3.4 we can see visually how different the diversity can be between the physical and routing layers of the same network, as is shown numerically in the following tables. Table 3.1 shows these properties for each network, as well as a number of standard graph metrics. For interactive visualizations of these topologies please refer to our web-based network mapping tool KU-TopView [218].

KU-TopView is a web-based tool we developed for several reasons. It allows for visual inspection of our hand-input topology data sets, in order to more-easily detect errors. It also facilitates easy viewing of all the topologies in our library both for us and for those

interested in our research. Thirdly it facilitates comparison of networks by overlaying one on another, and allowing selection of colors and other visual characteristics. Lastly it is able to combine networks and export both the original adjacency matrices and the combined matrices for use in other tools.

As will be seen in more detail in the results following, the characteristics of the topology significantly affect the diversity that can be attained. As mentioned earlier we assume the presence of a path server or equivalent service to provide the set of paths from which the selection is made. To perform this function we implemented a variation of the maximally diverse edge-disjoint path algorithm from [200]. We implemented a MATLAB simulation to evaluate the diversity measure and necessary supporting functions. Before examining the performance of path diversification with respect to flow reliability and stretch, we will look at its ability to improve the useable diversity of the graph as a whole.

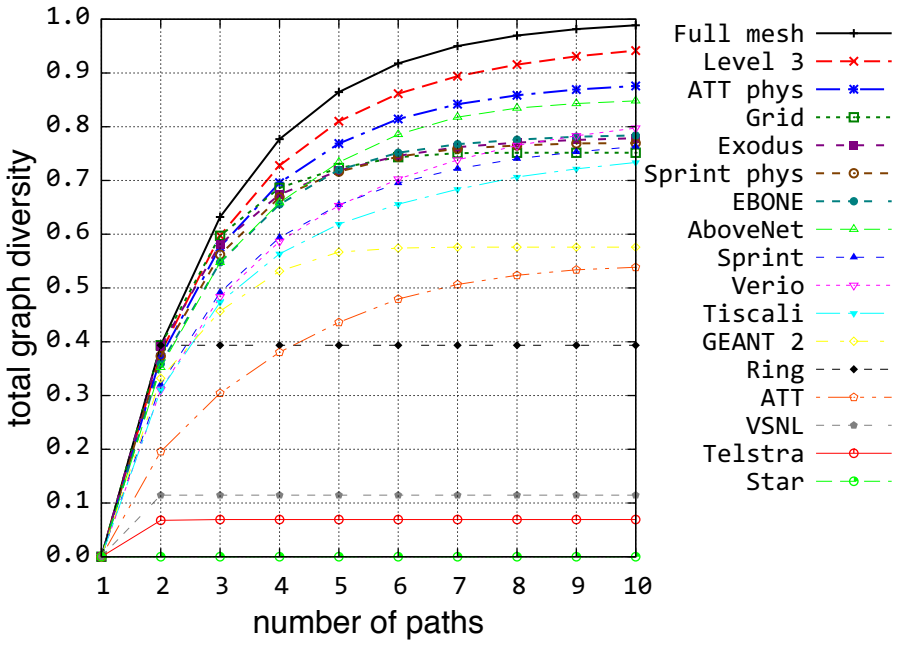

Figure 3.7: Total graph diversity vs. number of paths selected

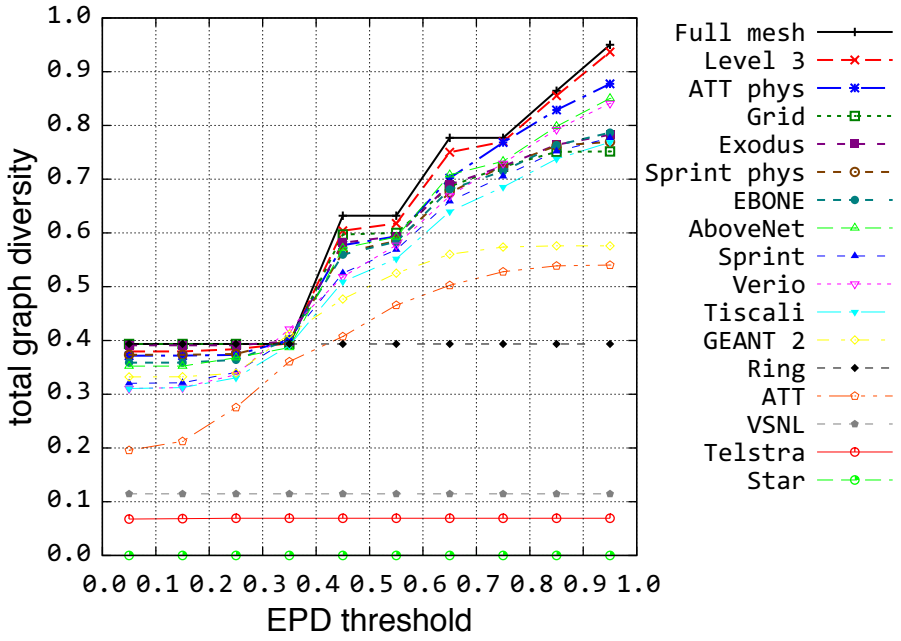

Figure 3.8: Total graph diversity vs. effective path diversity threshold

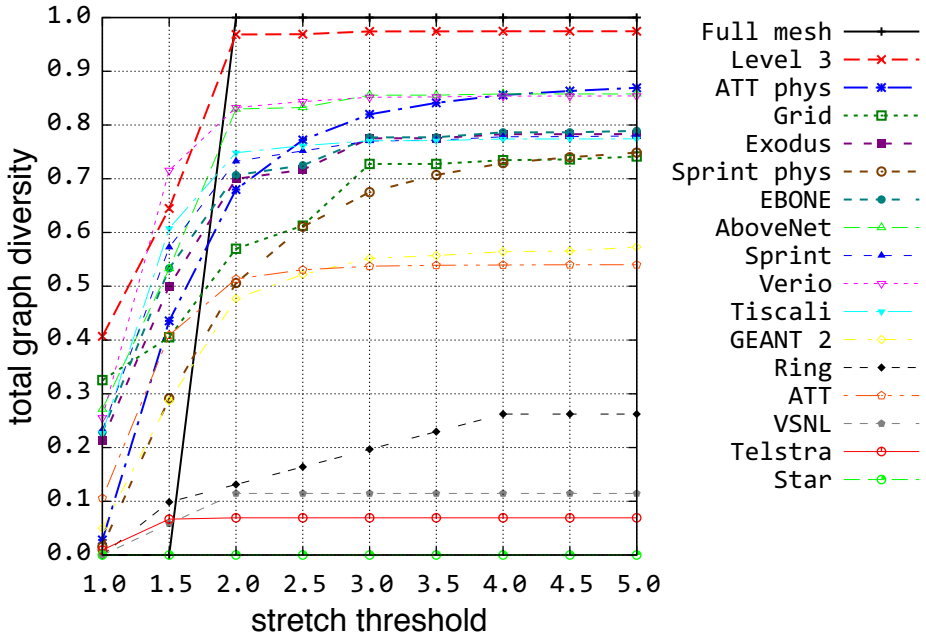

Figure 3.9: Total graph diversity vs. path-stretch limit

3.3.1 Diversity

By applying the diversity measure to the selected path set of an entire graph, we observe the effectiveness of the path diversification process. To do this we select a set of paths based on a given k or EPD threshold, and then calculate the TGD based on that set of paths.

In Figures 3.7, 3.8, and 3.9 we observe that most of the 17 topologies share similar characteristics in terms of the diversity available, although the absolute value of diversity that can be attained varies. Selecting k most diverse paths between nodes in the AT&T and GÉANT2 graphs results in a strong increase in diversity for $k < 4$ with limited improvement for additional paths, as we observe in Figure 3.7, however the full-mesh and Level-3 graphs continues to show substantial improvement for $k < 6$. This emphasizes the fact that basing a diversity metric on a particular number of diverse paths is highly topology dependent. Figure 3.8 gives the TGD with respect to EPD, which shows similar relative diversity values for the most of the topologies, but compresses many of the curves into a smaller vertical region, due to the fact that the measure itself is adaptive to the topology. From Figure 3.9 we observe that in all cases, virtually no additional diversity is gained beyond a stretch threshold of 3.

3.3.2 Dependability

A flow is established between each node pair using a set of paths determined using the path diversification algorithm and the specified diversity threshold value. To simulate link failures we remove each link from the graph based on a fixed probability of failure. In these simulations we use a uniform probability across all the links in the graph, however if we were to gain access to more detailed information on the network properties we could instead use an annotated graph to differentiate between the failure probabilities of the

individual links. To simulate node failures we remove all links connected to a particular node based on a fixed probability of failure for that node. A flow is considered *reliable* if at least one of its paths remains unbroken by the link or node failures. We compute *flow robustness* to be the number of reliable flows, divided by the total number of flows in the network. For each probability of failure, we also determine the the best flow robustness possible for any path-selection mechanism given the partitioning of the underlying graph, and show this value in each plot with the label 'Best'. Finally, we, calculate the flow robustness using only the conventional shortest path for each node pair to serve as a lower performance bound.

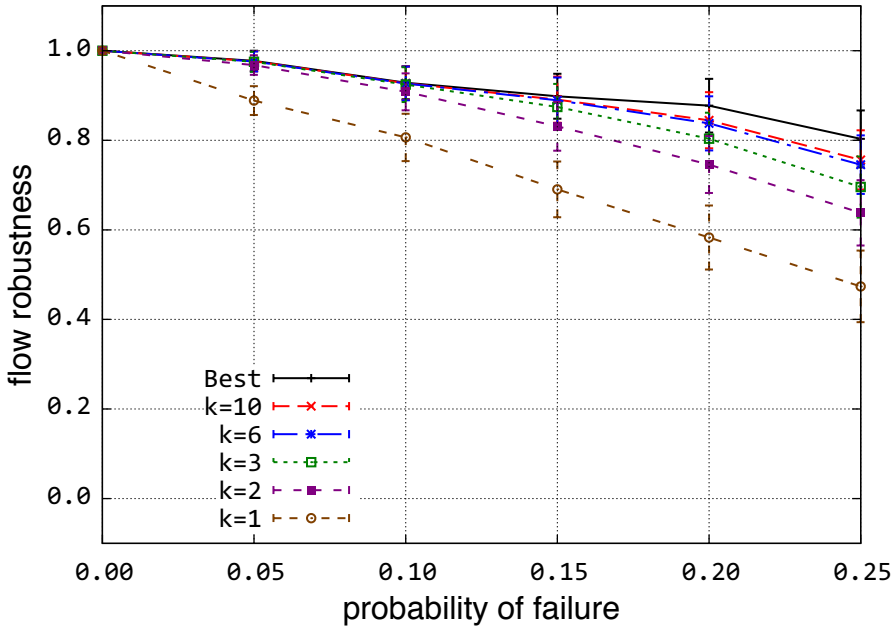

Figure 3.10: Flow robustness vs. link failure probability for the Sprint logical topology

To accomplish all this, we start with the set of paths between all node pairs in the network selected by path diversification to meet a particular path-diversity threshold. We then remove each each edge of the graph independently with probability p. After calculating the resulting flow robustness we reset the graph and repeat the process 100 times for each

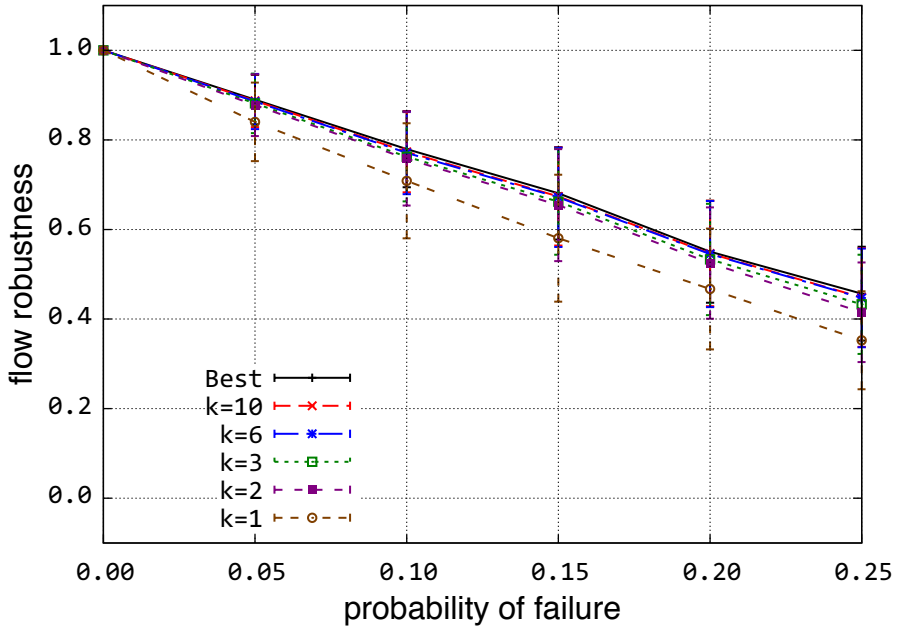

Figure 3.11: Flow robustness vs. node failure probability for the Sprint logical topology

probability p before continuing to the next path-diversity threshold. Each point plotted is the average of these 100 trials, and the error bars represent 95% confidence intervals.

Taking the Sprint logical topology as an example, we show the flow robustness in the face of random link failures (Figure 3.10), node failures (Figure 3.11), and combined link and node failures (Figure 3.12). In the link-failure case we can clearly see that the improvement from $k = 1$ to $k = 2$ is the most significant (>30% improvement with a 25% probability of failure), and that with $k = 6$ we are getting pretty close to the upper bound of the graph connectivity (represented by the 'Best' curve on the plots). In the node-failure case we note 3 points of interest. First the confidence intervals are much larger, due to the fact that on average a singe node affects more flows than a single link resulting a coarser granularity of simulation results and more variance from one run to the next with the same probability of failure. Secondly, the network connectivity drops much more rapidly than it did with only link failures, with only 40% connectivity with a

25% failure rate, as opposed to the 80% connectivity with only link failures. Thirdly we see that there is much less room for improvement between $k = 1$ and the upper bound, and that $k = 2$ closely approaches the upper bound. When we combine node and link failures, the network connectivity is only slightly worse than with node failures alone, however the single-path case drops nearly 50%, to less than 20% of flows surviving at a 25% failure rate.

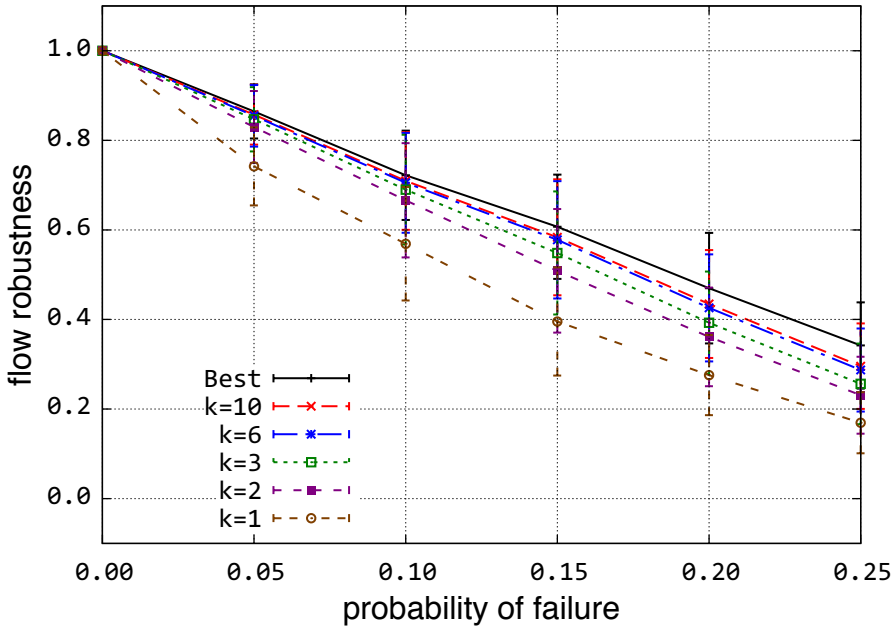

Figure 3.12: Flow robustness vs. node & link failure probability for the Sprint logical topology

To look at the dependability of a few networks with significantly different properties than the Sprint logical topology, we have selected the Sprint physical network, the Level-3 logical topology, and the VSNL logical topology, the maps of which were shown previously. We see that the network connectivity of the Sprint physical network drops fairly quickly, and in this case even selecting 10 paths is not sufficient to utilize all of the diversity available in the network. This may be due in part to the Floyd-Warshall heuristic algorithm being used to populate our path database and we expect that the results would be

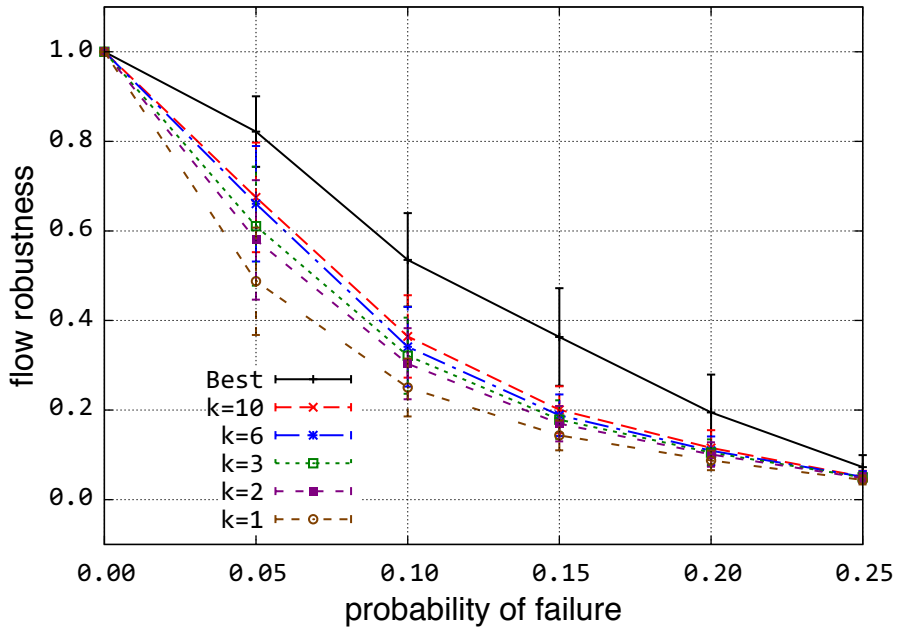

Figure 3.13: Flow robustness vs. node failure probability for the Sprint physical topology

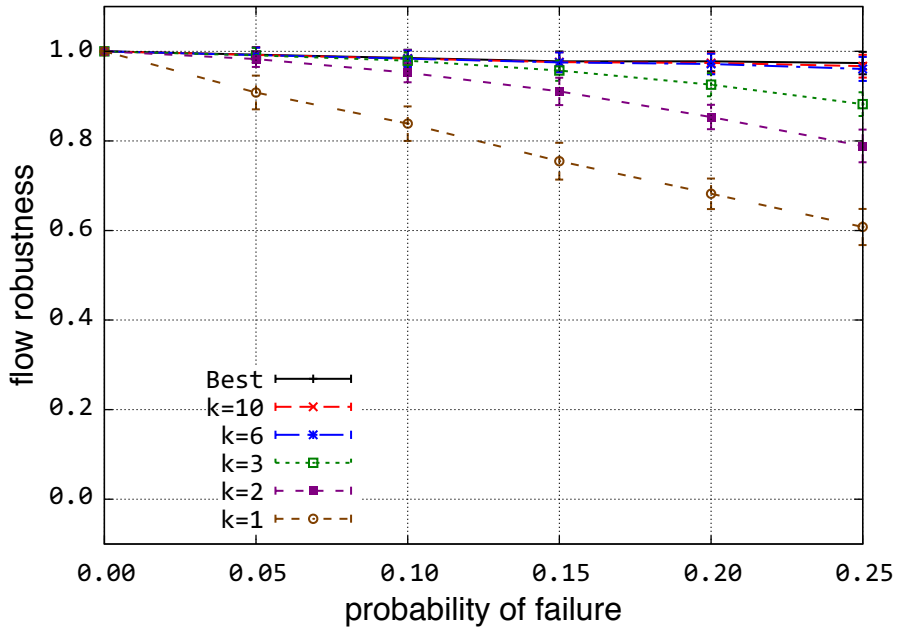

Figure 3.14: Flow robustness vs. link failure probability for the Level-3 logical topology

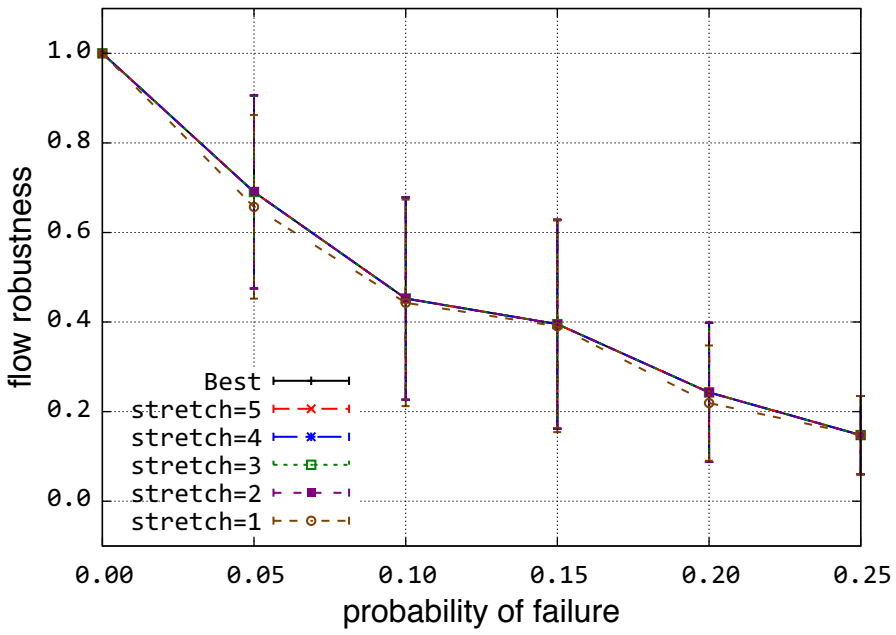

Figure 3.15: Flow robustness vs. node & link failure probability for the VSNL logical topology

improved when using a distributed exhaustive path discovery algorithm. In looking at the Level-3 network with link failures, we see that its high degree of connectivity keeps nearly 100% connected throughout the range of failure probabilities simulated, however using a single-path algorithm still results in 40% of the flows being disrupted. Choosing six paths approaches the upper bound of the network's capabilities. At the other end of the spectrum, the VSNL is very poorly connected, resulting in a rapid drop in connectivity as links and nodes fail, and a situation where there are almost no alternate paths so path diversification provides almost no improvement over single-path routing. Plots showing the flow robustness results for all 17 topologies may be found in Appendix E.

Resilience Improvement

By using the ResiliNets resilience framework [191, 192, 219–221], we can show that path diversification does improve the resilience ($\mathbb{R}$) of the network. The resilience framework

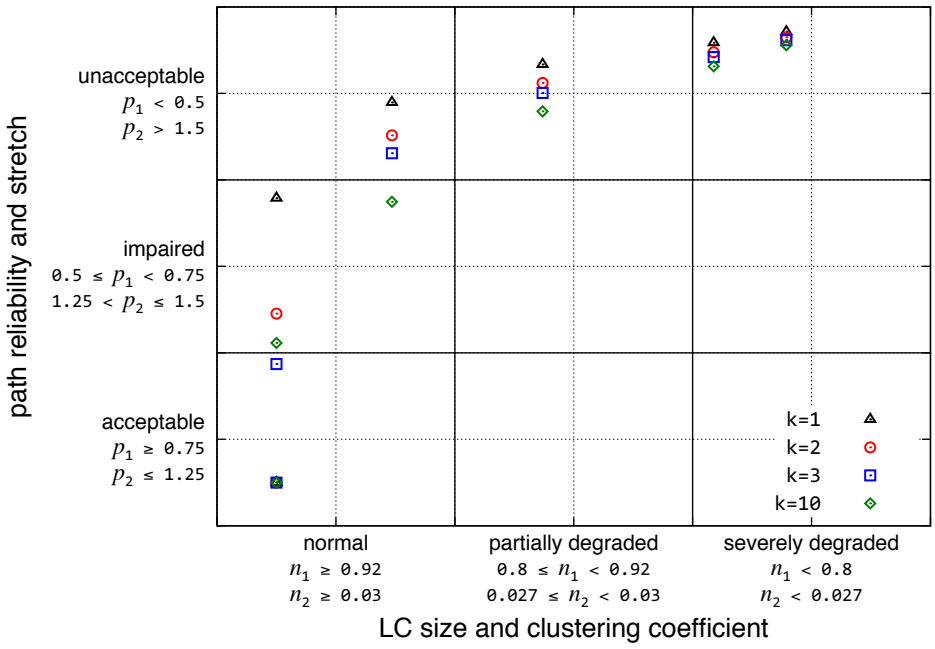

Figure 3.16: Resilience of path diversity for the AT&T physical topology

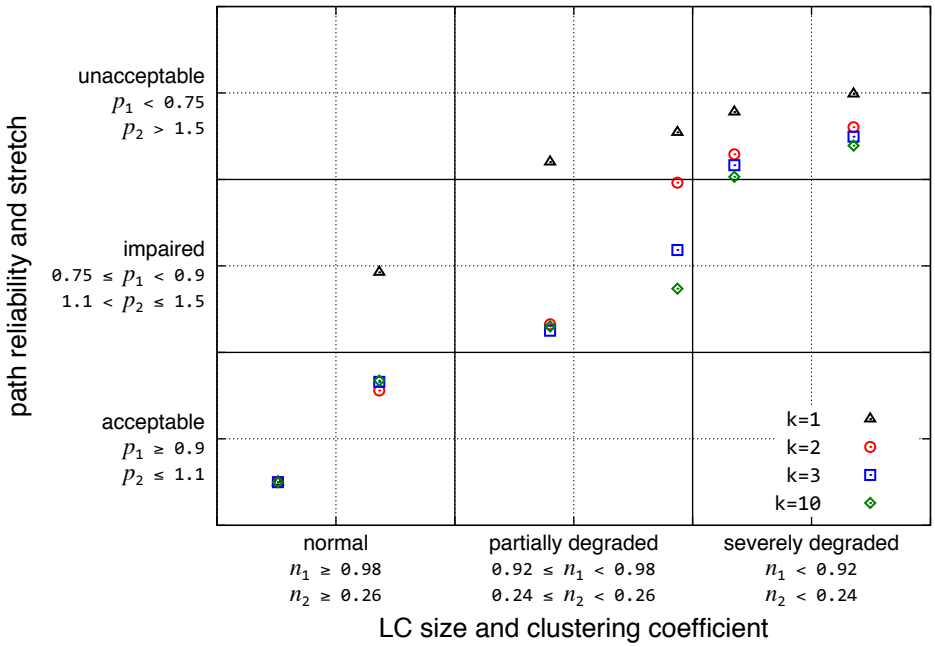

Figure 3.17: Resilience of path diversity for the AT&T physical topology

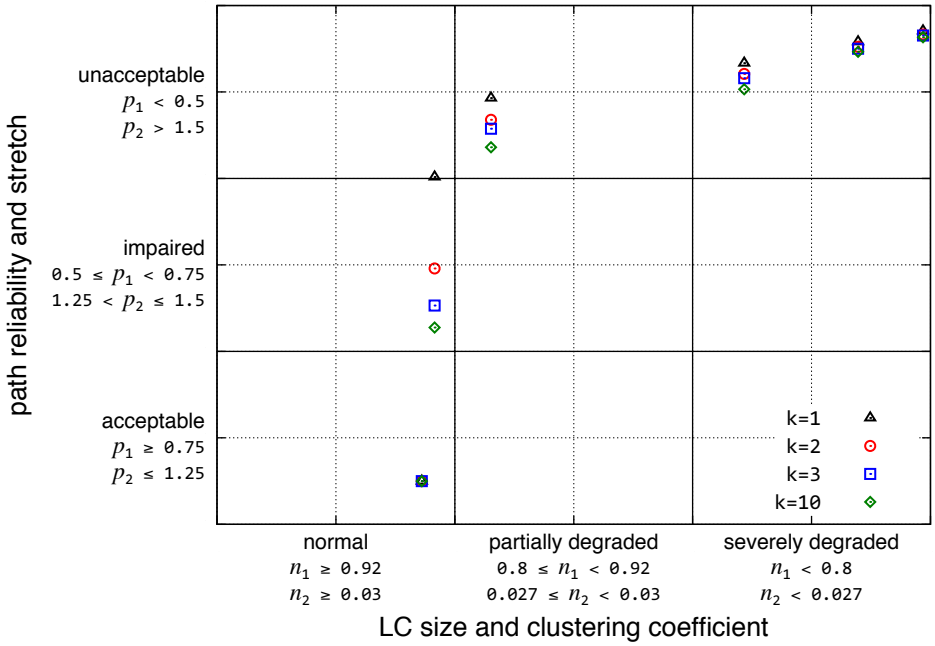

Figure 3.18: Resilience of path diversity for the AT&T physical topology

allows us to take multiple properties of the network operational conditions (shown on the x-axis), and show the corresponding states of the service being provided at a particular layer boundary. In this case we are interested in the condition of the network topology, capture using the largest component size (LC size) and clustering coefficient. The service being provided is end-to-end connectivity, the state of which is captured using path reliability and average stretch.

Table 3.2: Aggregate resilience of path diversity

Number of paths	Aggregate Resilience $\mathbb{R}$			
	$k = 1$	$k = 2$	$k = 3$	$k = 10$
AT&T	0.38858	0.43021	0.45076	0.47638
GÉANT2	0.48909	0.62584	0.64619	0.65773
Sprint	0.40013	0.42458	0.43445	0.45063

Plots resulting from this framework show the normal state (normal network conditions and acceptable service performance) near the origin, with network conditions degrading

to the right on the x-axis and service performance degrading upwards on the y-axis. A resilient network is one that stays in the normal operating condition while undergoing challenges, and a resilient service is one that stays in the acceptable service state as the network degrades. We quantify the degree to which a service degrades as the network operating coditions degrade using the area under each curve, and quantify $\mathbb{R}$ as the total area minus the area under the curve in question, normalized on a scale from zero to one.

We use the AT&T, GÉANT2, and Sprint physical-layer networks as examples, as shown in Figures 3.16, 3.17, and 3.18. For each network we plot services using 1, 2, 3, and 10 paths, and note that in each case using a greater number of diverse paths results in a service that stays closer to the x-axis as network conditions degrade. We see the most dramatic improvement in the case of the GÉANT network, while in the cases of AT&T and Sprint rapidly increasing stretch limits the performance improvement available through path diversity. Table 3.2 quantifies the improvement in $\mathbb{R}$ aggregated across the full range of network conditions analyzed.

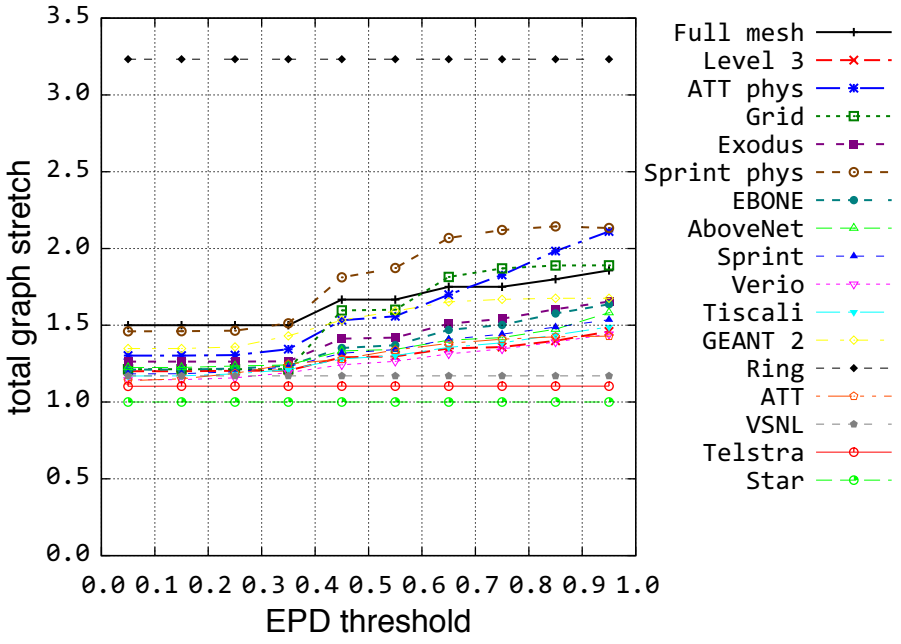

Figure 3.19: Total graph stretch vs. effective path diversity threshold $\lambda = .5$

3.3.3 Path Stretch

By definition, if we are using paths other than the shortest-path as determined by conventional routing mechanisms, those paths will be longer, and this is measured in terms of *path stretch* as defined previously. Depending on the particular topology in use, and the source-destination pair being measured, some paths will be several times as long as the original, while others will be only slightly longer, and a particular flow will be composed of paths with varying stretch. To evaluate the effect of path diversification at the graph level we define *average path stretch* (APS) to be the average stretch over all selected paths between a give source-destination pair. The *total graph stretch* TGS is then the average of the APS values for all node pairs in the graph. We can then plot TGS with respect to EPD to determine the cost of increased diversity in terms of path stretch. As seen in Figure 3.19, the response of TGS to increasing EPD threshold is significantly topology dependent. The TGS of the ring is virtually unchanged by increasing EPD, due to the existence of only 1 alternate path, which on average is significantly longer than the primary path. The TGS of several topologies increases at a relatively linear rate, indicative of the relatively high degree of connectivity in those networks. In contrast the Sprint physical and full mesh have distinct thresholds near EPD values of 0.4 and 0.6, due their particular characteristics. In most of the networks we observe that the stretch remains less than double even for very high EPD thresholds.

While it is observed that stretch is likely to increase when selecting diverse paths, we want to observe the relationship between diversity and stretch. To this end we calculate a *gain*, which is the ratio of TGD divided by TGS for a particular requested EPD threshold. This normalizes the diversity improvement seen to the increased cost in terms of stretch. Figures 3.20, 3.21, and 3.22 quantify small increases in path diversity result in a large improvement in overall graph diversity, while incurring minimal stretch, however this

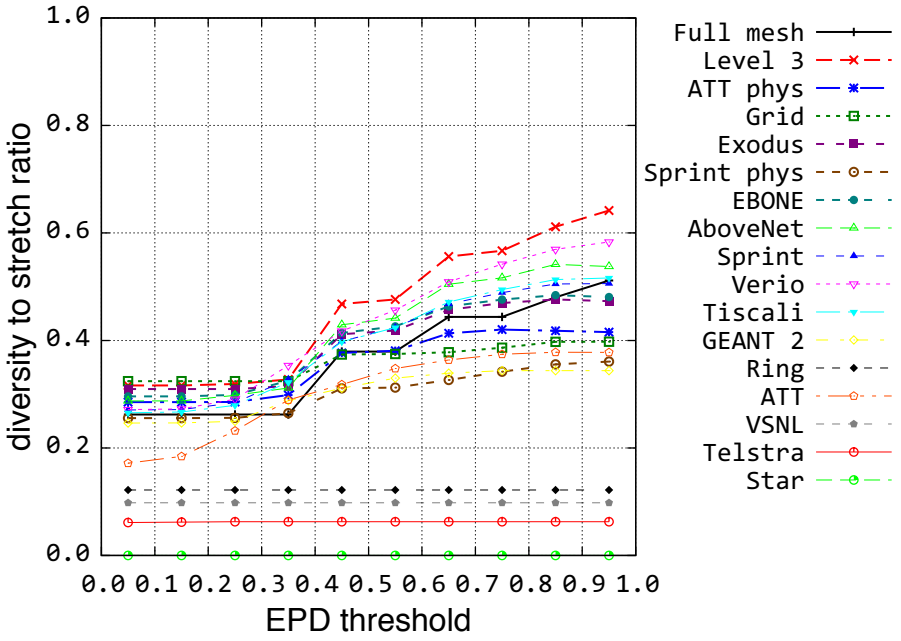

Figure 3.20: Gain vs. effective path diversity threshold where $\lambda = 0.5$

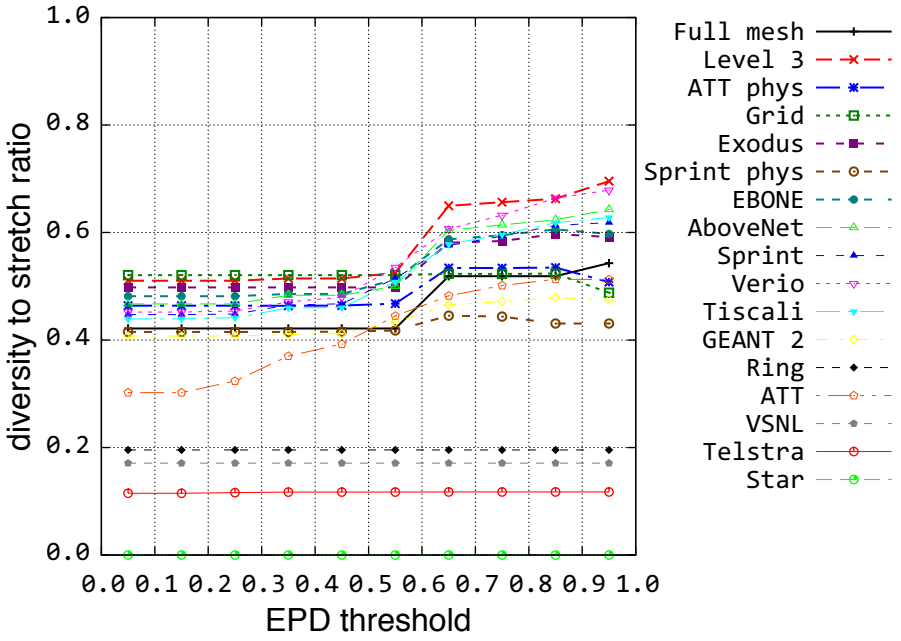

Figure 3.21: Gain vs. effective path diversity threshold where $\lambda = 1$

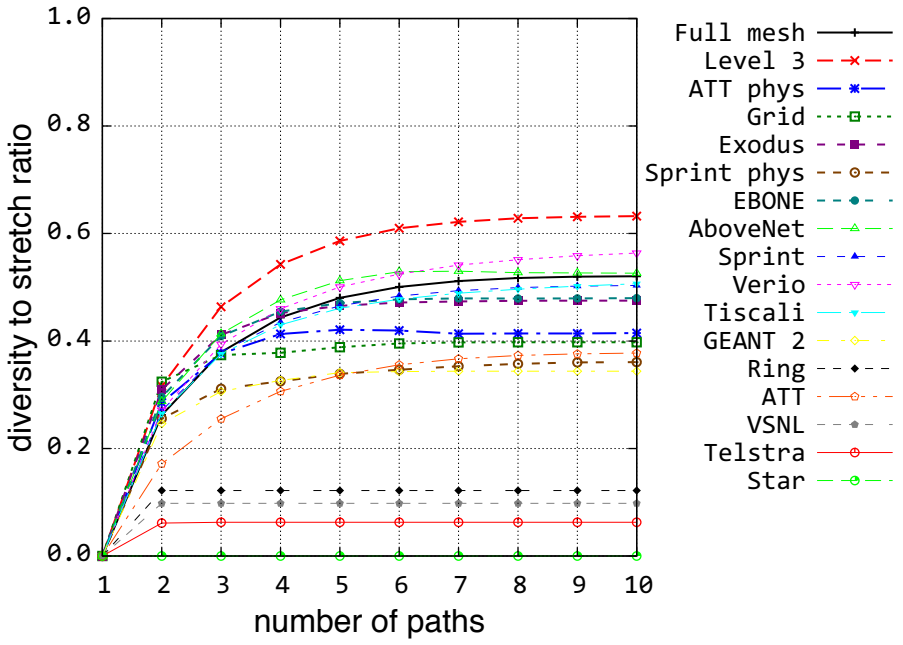

Figure 3.22: Gain vs. number of paths requested

effect is limited by the underlying topologies such that when no new diverse paths are available the ratio becomes constant. Networks with little diversity, or a large amount of stretch appear at the bottom of this plot, namely Star, Telstra, VSNL, and Ring. Comparing Figure 3.20 and Figure 3.21 we observe the effect of modifying λ. When λ is increased, the improvement in diversity does not come until higher EPD thresholds (due to the increased impact of k on EPD), however the overall gain is higher. Figure 3.22 shows the same gain measure when plotted against k, and again emphasizes the role played by the underlying topology. In this case the Sprint topology is still showing improved gain well after the other two have flatlined.

Here we note that a multipath-aware application would likely not use all reliable paths equally, so averaging the stretch of all paths overestimates the effective stretch in that case.

3.4 Topology Survivability Comparison

In comparing the survivability of various topologies, we are concerned not with the performance of a particular protocol or mechanism in recovering from failures, rather we are considering the survivability inherent in the structure of the topology itself. To do that, we calculate the *flow robustness* as the probability of link and node failures is increased. In this case we are considering a flow to be in tact as long as a path exists that connects the source and destination, i.e. the source and destination nodes are not partitioned from one another. Questions of reconvergence time and path protection accuracy are all protocol specific and outside the scope of this work. We compare 17 topologies, 4 of which are synthetic topologies included for completeness. Ten topologies are logical router-level topologies inferred by the Rocketfuel project [222]. The remaining three are physical-layer fiber topologies based on [223]. Figures 3.4 and 3.3 show the physical and logical-layer topologies respectively for the Sprint network, as an example of the substantial differences between these two categories of maps. The synthetic topologies can be easily recreated based on the data in the tables below, and the data (including adjacency matrices and node geo-locations) for the real topologies is available via KU-TopView, our Web-based topology map viewer [218].

3.4.1 Simulation Results

To perform the failure simulations we use MATLAB since we are not looking at dynamic or transient behavior and therefore do not need to simulate packet flows but can perform the calculations using graph-theoretic methods. For each of the 17 topologies we use the following process:

1. load topology adjacency matrix

2. calculate 300 failure sets based on current probability

   (a) 100 sets with link-failures only

   (b) 100 sets with node-failures only

   (c) 100 sets with link and node failures

3. calculate fraction of node-pairs connected in each set

4. average across each 100 sets

5. plot 3 data points

6. increment failure-probability until all values range are complete

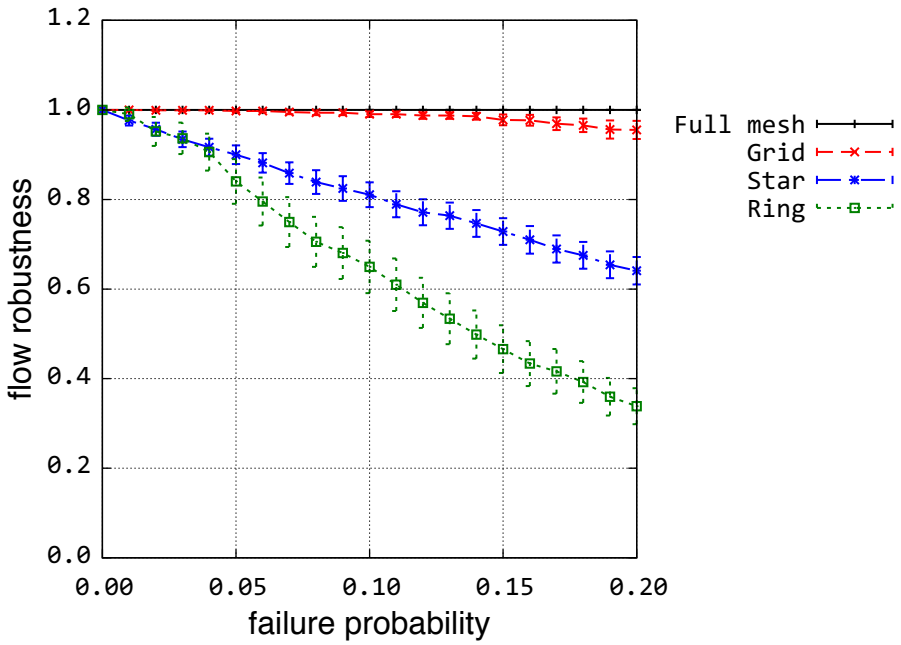

Figure 3.23: Link failure robustness (synthetic)

For this evaluation we use 51 failure probabilities evenly distributed over the range 0–0.5 inclusive, resulting in 15,300 simulation runs for each topology, or 260,100 runs total,

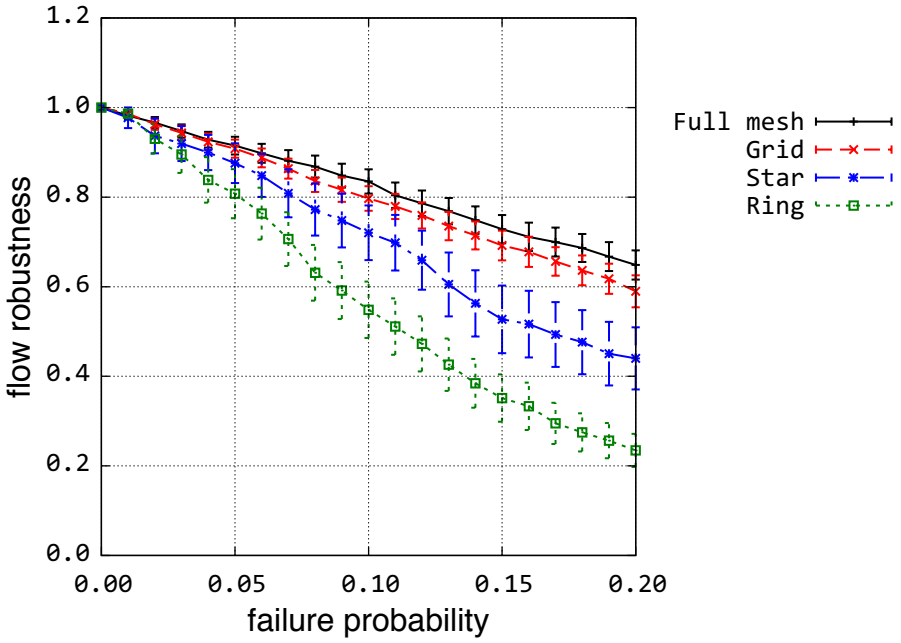

Figure 3.24: Node failure robustness (synthetic)

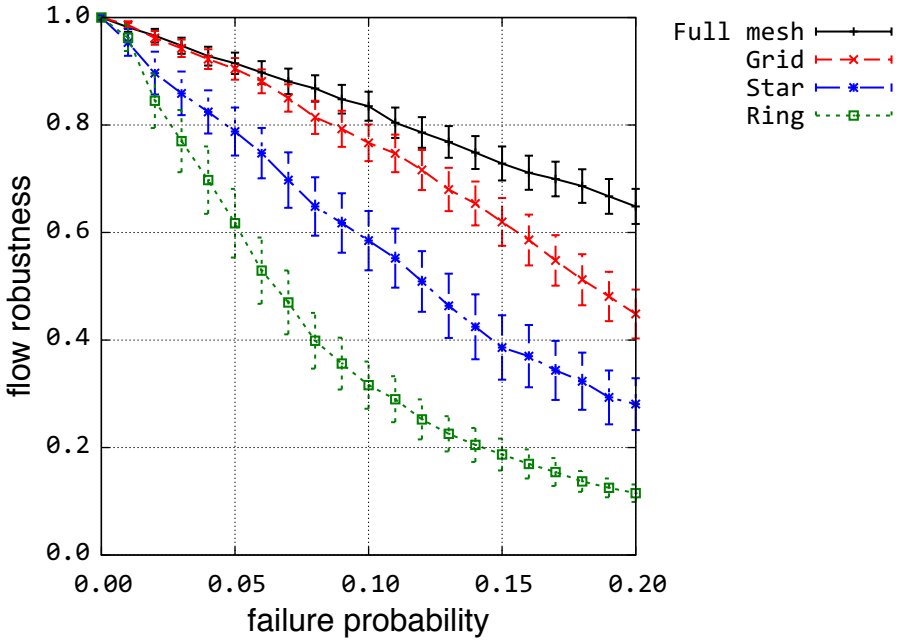

Figure 3.25: Node & link failure robustness (synthetic)

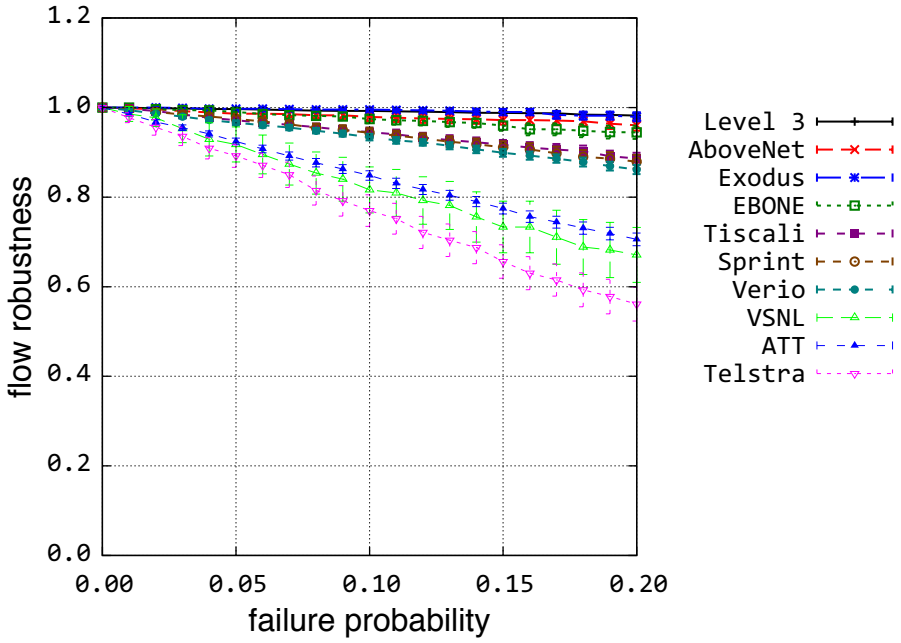

Figure 3.26: Link failure robustness (logical)

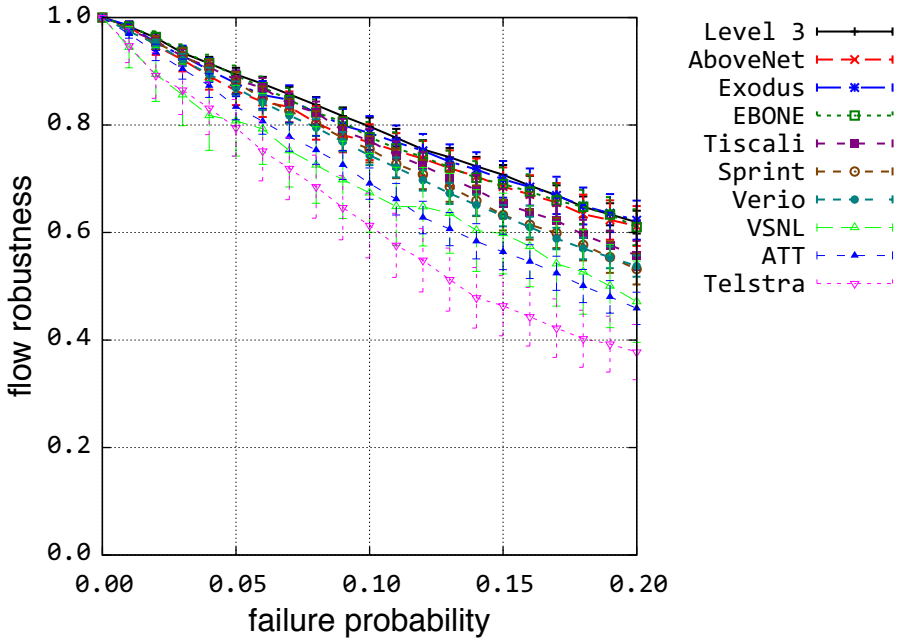

Figure 3.27: Node failure robustness (logical)

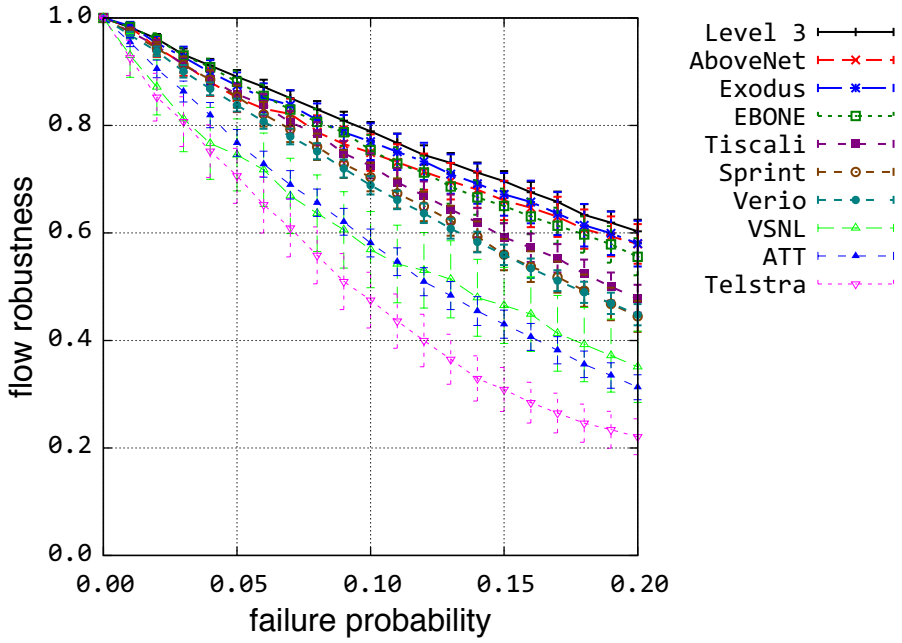

Figure 3.28: Node & link failure robustness (logical)

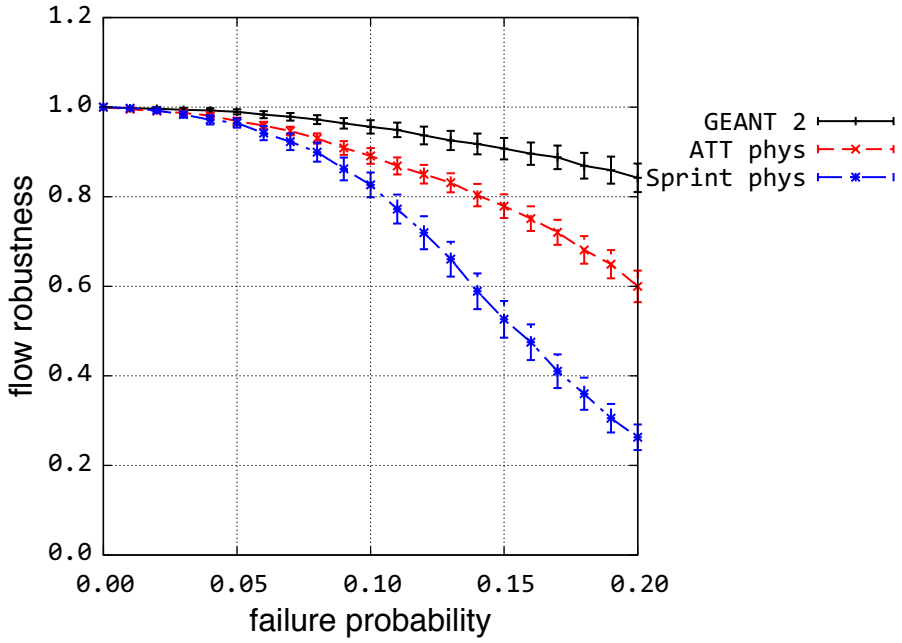

Figure 3.29: Link failure robustness (physical)

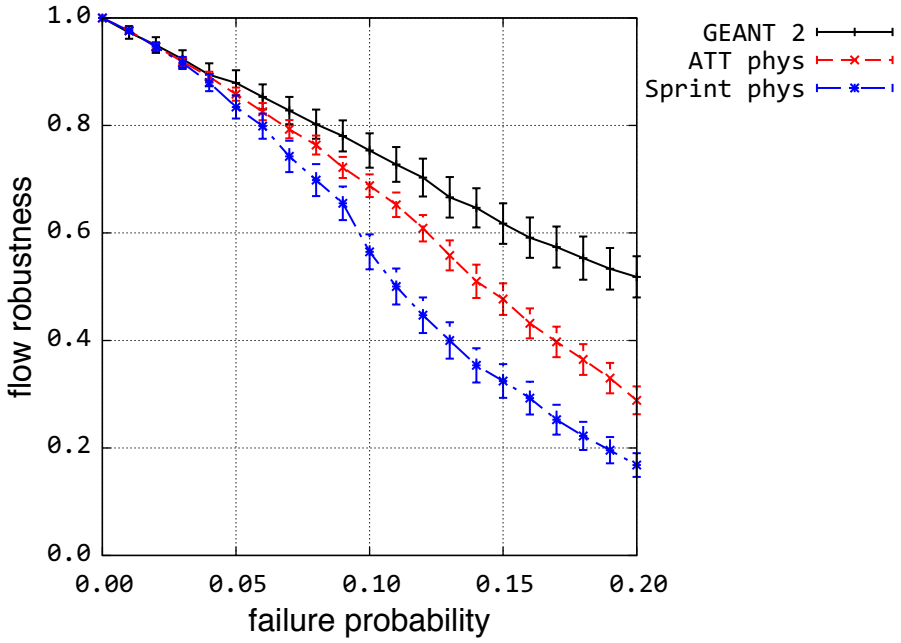

Figure 3.30: Node failure robustness (physical)

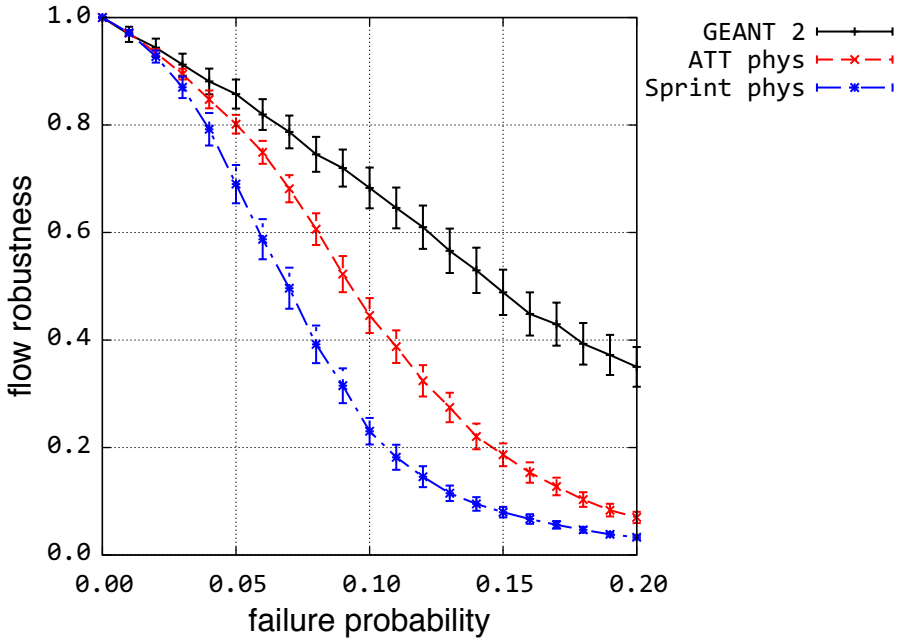

Figure 3.31: Node & link failure robustness (physical)

which took several days to complete using a computing cluster consisting of approximately 1000 Intel Xeon CPU cores. The results of this process are summarized in Figures 3.23–3.31. These plots are a collection of the *best-possible* or *reference* curves that would appear on a plot comparing routing or path protection schemes. The curve for each plot is distinct due to its topology, and thus from these plots we can quickly see which ones are more survivable than others. We have separated the plots into three categories: logical, physical, and synthetic, for plotting purposes simply for readability because they became difficult to distinguish with too many curves in each plot. At this point we can also take note of specific topology's performance, for example the full-mesh does best overall, while the ring is the worst of the synthetic topologies, and the Sprint and AT&T physical topologies do the worst overall. What we see from these plots is that the relative ordering of the curves remains largely unchanged (the few exceptions are of minimal size) and so it is reasonable to expect that a measure of survivability may be computed based on the topology alone, without being dependent on the expected probability of failure of individual links and nodes. For comparison purposes we plot the effects of node and link failure on a number of graph properties besides the connectivity (flow robustness) plots shown here. The additional plots are in Appendix D.

3.4.2 Topology Characteristics Survey

Table 3.1 lists all of the topologies analyzed, along with a set of standard graph-metrics defined as follows:

- Node degree: "The number of connections or edges the node has to other nodes." [224]
- Bi-connected: A graph in which the minimum node degree is two or greater.
- Clustering coefficient: "A measure of how many nodes form triangular subgraphs with their adjacent nodes." [225] or "The fraction of paths of length two in the

network that are closed" [226] which may be interpreted as "A measure of degree to which nodes in a graph tend to cluster together." [227]

- Diameter: The maximum shortest-path between any node-pair.
- Radius: The minimum of the maximum shortest-path for all nodes.
- Hop-count: The *average* shortest-path between all node-paris.
- Closeness: "The mean distance from a vertex to other vertices." [228] Closeness is a measure of centrality and is related to node degree.
- Betweenness: "Betweenness is the number of shortest paths passing through a node or link and provides a centrality or importantness measure." [229, 230]

For each metric, the best (w.r.t. survivability) three values are highlighted in bold font. A number of these features are linked to network resilience in one way or another, for example topologies with a high average node-degree generally have more protection paths available, while topologies with a high maximum node or link betweenness may have a central point-of-failure which would be targeted by an attacker. Diameter, radius, and hop-count are closely related distance metrics. In our preliminary work on *path diversification* [212] (with a much smaller sample-set of topologies available to work with) it became apparent that the TGD metric was able to differentiate similar topologies according to their survivability performance, but looking at the plots and Table 3.1 it is clear that this no longer holds true when the network size varies widely. None of the listed graph theory metrics (or any we are aware of) correlate closely to network survivability as simulated in Section 3.4.1. A major contribution of this dissertation is a new metric that closely correlates with survivability, which we present in Section 3.5.

3.5 Analysis

In this section we further explore the relationships between the metrics listed in Section 3.4.2 and topology survivability.

Table 3.3: Network Rank

Network	Survivab. Rank	Node Deg. Rank	TGD Rank	Clustering Rank	Diam. Rank	Hopcount Rank	Closeness Rank	Node Bet. Rank	Link Bet. Rank
Full-Mesh	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Level 3	**2**	**2**	**2**	**2**	4	**2**	**3**	10	9
AboveNet	**3**	**3**	8	**3**	**3**	**3**	**2**	5	**3**
Exodus	4	5	5	8	4	5	6	4	4
EBONE	5	5	7	10	4	7	7	4	6
Tiscali	6	4	11	4	5	8	8	9	10
Sprint	7	5	9	6	5	9	9	8	11
Verio	8	4	10	7	7	10	10	13	13
Grid	9	6	4	15	7	12	12	**3**	7
VSNL	10	7	15	5	4	6	5	**2**	**2**
GÉANT Phys.	11	6	12	11	8	14	14	7	12
Star	12	8	17	15	**2**	4	4	6	5
AT&T	13	7	14	9	6	13	13	14	15
Telstra	14	7	16	12	6	11	11	11	14
Ring	15	7	13	15	9	15	15	4	8
AT&T Phys.	16	7	**3**	13	10	16	16	15	17
Sprint Phys.	17	7	6	14	10	17	17	12	16

3.5.1 Topology Ranking

Based on the flow robustness results (Section 3.4.1) we can rank the topologies based on their survivability in the presence of multiple failures[1] as shown in Table 3.3. This ranking can easily be done qualitatively by visual inspection of the plots above, with higher curve outranking lower curves. To perform the ranking a bit more rigorously we chose a fixed point on the x-axis (0.2 in this case) and ranked each topology according to its value at that point on the link and node failure plot. Since the curves have minimal crossover, this produces the same ranking as the visual inspection approach. The metric values

[1]This is not to serve as a recommendation of one network over another for business purposes. Due to common business practices the Internet service providers listed (with the exception of GÉANT2) do not make their network topology data publicly available. The data sets used are inferred by third parties and are over 10 years old in some cases.

from Table 3.1 are also shown here as rankings in order to emphasize the correlation (or lack thereof) between each particular metric and the survivability rank. Some metrics are not included in this table, for example the number of nodes and links that are a direct measure of the graph size, and are not a unique property of the topology design, and the *radius*, which is so closely related to the diameter and the hop-count as to be redundant. In contrast, the average node degree relates the number of nodes to the number of links.

From Table 3.3 we see that most of the metrics correctly rank the top 2 or 3 networks according to their survivability, but beyond that the rank no longer corresponds. Based on previous experience with the *path diversity* metrics, and the intuition that diversity should be closely correlated with survivability, we investigated further and developed the *Compensated Total Graph Diversity* metric.

3.5.2 Compensated Total Graph Diversity

In Section 3.2.5 we noted that the *diversity* metric is independent of path length, meaning that there is no natural penalty assigned to longer paths as opposed to short ones. On the other hand, there *is a significant statistical penalty to long paths* when simulating probabilistic failures. Intuitively this penalty results from greater exposure in real networks to component failure due to natural faults or intentional attack. Returning to Table 3.1 we see that specific topologies receive a much higher TGD-rank than survivability-rank (e.g. AT&T Physical, Sprint Physical) also have much higher diameters than other networks with similar TGDs. Conversely, the star topology, which is given the lowest TGD rank but performs better than 5 other networks, has a much smaller diameter than the networks it outperforms. Further investigation shows that the average hop-count is a more precise indicator of this penalty than the diameter or radius. Based on this we propose a new composite metric that takes into account *both* TGD and average hop-count.

We call the new metric *Compensated Total Graph Diversity* (cTGD) and define it as follows:

$$\text{cTGD} = e^{\text{TGD}-1} \times h^{-\alpha} \tag{3.13}$$

where h is the average hop-count and α is a parameter tuned experimentally. Increasing α reduces the negative impact of increasing hopcount on cTGD. By trying several values between one and two we find that $\alpha = 1.25$ gives the best correlation to our simulation results. Other values resulted in changes at the fourth decimal place of the cTGD, which is enough to cause some reordering in the topology rankings. The benefit of taking the exponential of the original TGD is that the range is still bounded between 0 and 1, but is no longer inclusive of 0, which allows for the cTGD of a topology with 0 diversity to be positive, as in the case of the Star network. From the hop-count component we desire an inverse relationship (higher hop-counts result in lower cTGDs), and we use the α parameter to tune the aggressiveness of this relationship. To put equation 3.13 in the context of providing an end-to-end service, it requires greater diversity to provide a given level of flow reliability over a long path than is required to achieve the same level of flow reliability over a short path. In the graph context, a large-diameter graph must provide a higher TGD to achieve the same level of flow-robustness as a smaller-diameter graph with a lower TGD.

Table 3.4 again shows the topologies ranked according to their simulated survivability results, alongside their cTGD metric value and cTGD rank. We see that both measures provide an identical ranking for all the topologies suggesting that the cTGD metric is an excellent predictor of topology survivability. We note here that we are not claiming that this *exact* correlation would hold true for *every* possible set of topologies, only that we expect a close correlation. The reason for this is that the TGD is a heuristic

Table 3.4: Compensated TGD

Network	**Survivability Rank**	**cTGD**	**cTGD Rank**
Full-Mesh	**1**	**0.9514**	**1**
Level 3	**2**	**0.4494**	**2**
AboveNet	**3**	**0.4386**	**3**
Exodus	4	0.3617	4
EBONE	5	0.3113	5
Tiscali	6	0.2641	6
Sprint	7	0.2407	7
Verio	8	0.2009	8
Manhattan Grid	9	0.2002	9
VSNL	10	0.1783	10
GÉANT2 Phys.	11	0.1668	11
Star	12	0.1628	12
AT&T	13	0.1446	13
Telstra	14	0.0941	14
Ring	15	0.0667	15
AT&T Phys.	16	0.0348	16
Sprint Phys.	17	0.0307	17

measure, and the survivability rank is based on a Monte Carlo simulation set, both of with introduce a margin of error. We have sought to reduce this error as much as possible through our methodology. That being said, what we are seeing is a strong correlation so that any reordering should only occur when two topologies have *very* similar cTGD and survivability metric values to begin with. This implies that cTGD alone may be used as a quantitative indicator of a topology's survivability, without performing extensive failure analysis as we have done in Section 3.4.

3.6 Summary

This chapter introduced *path diversification*, presenting its design, evaluation and use as a topology analysis tool. We also discussed several metrics for evaluating path, node-pair, and graph diversity, and applied the path diversification mechanism to 17 real and synthetic networks and evaluated its ability to improve flow robustness in the presence of link and node failures. In the next chapter we will begin by looking at the ResTP

transport protocol, in which path diversification is the core component that enables end-to-end multipath support.

Page left intentionally blank.

Chapter 4

Transport Protocol Design

Mechanisms are the building blocks that make up each layer of the network stack, including transport protocols. The selection of mechanisms determines the resilience of the protocol, so we use the ResiliNets Principles to guide this selection process. Some mechanisms are basic to data transfer, and some address particular network characteristics or challenges. There are also cross-layer mechanisms that provide the vertical interconnection between layers of the network stack.

Closely tied to the selection of mechanisms is the principle of *tradeoffs*. For each mechanism added to a protocol there is an increase in complexity. Depending on the particular mechanism there may also be a cost in terms of latency, jitter, or bandwidth. Some mechanisms provide overlapping functionality but differing performance, making it necessary to select between them. Because of these tradeoffs it is important to use mechanisms only where they yield the greatest benefit and are appropriate to the network state. Since this state may change dynamically, it may be necessary to tune mechanisms or even turn them on and off completely during a single transport session. Negotiating their use at session setup is not sufficient.

As stated in Section 1.2 our goal is to show that *E2E Communication with resilience as an inherent design property is necessary to meet specified service requirements in the*

face of various attacks and challenges and that *diversity is an essential characteristic of network topologies that enables enhanced resilience at the end-to-end layer.* A key aspect of this work is not only to explore, measure, and analyze alternatives at the transport level, but to understand the multilevel interactions and tradeoffs with the network layer. The goal is to provide *adaptability* by having some ability for the application layer to *tune* the resilience level of the lower layers.

In this chapter we present the design of two transport protocols which are closely related. First is the ResTP protocol, which is a multipath protocol based on the Path Diversification process presented in the previous chapter. It also uses multiple reliability paradigms in order to meet the service requirements of a variety of application in multiple environments. The second protocol is AeroTP, which uses a subset of the mechanisms present in ResTP to meet the requirements of the highly-dynamic airborne telemetry environment. AeroTP shares the multiple reliability-mode structure with ResTP, and adds the concept of custody-transfer using gateways, which also allows for translation to TCP and UDP. AerpTP does not make use of multiple paths, and so does not include the path diversification framework in its design. Due to the more focused nature of the AeroTP design requirements, we present it here as an in-depth case study to evaluate a subset of the ResTP functionality.

The work presented in this chapter has resulted in a number of publications. We originally proposed and evaluated the ResTP protocol in [211]. We proposed AeroTP in [231] and the ANTP suite as a whole in [232]. Since then we have regularly revised the protocols and expanded our analysis of them in [233–238]. The remainder of this chapter is organized as follows: Section 4.1 presents the intended function, protocol design, mechanism tradeoffs, and multipath simulations of the ResTP protocol. AeroTP follows, with Section 4.2 giving an overview of the ANTP suite, including an overview of the challenges posed to traditional routing and transport protocols by the airborne environment in Section 4.2.3.

We then present the performability improvements gained using the suite in Section 4.4.

4.1 Resilient Multipath Transport Protocol

In order to be adaptable, we have designed multiple mechanisms, including path diversification, into a transport protocol with configurable service requirements. This allows it to be tuned for a variety of applications and network environments.

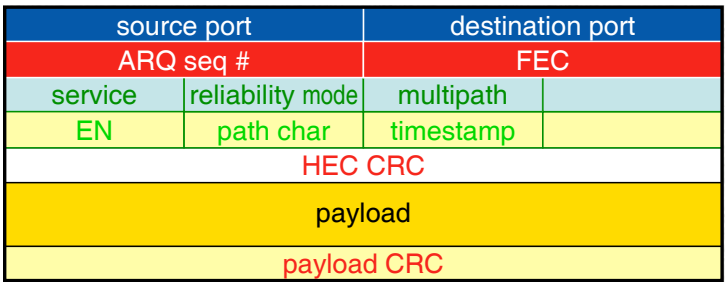

Figure 4.1: ResTP header showing knob and dial fields

4.1.1 Protocol Design

The ResTP header (Figure 4.1) is designed to support cross-layer parameters and shows the fields used to indicate the service parameters required to the lower network layers. This is supported at the PoMo gateways that exist at realm boundaries, such that we do not require every router on a path to interact with the ResTP cross-layer fields, however the PoMo gateways may interact with ResTP extensively in order to modify the ResTP behavior on a realm-by-realm basis. Based on the application requirements, there may be a number of data classes being transferred over the network. For this reason we define multiple *reliability modes* that are mapped from different service types and for the generic counterpart of the AeroTP reliability modes [231]. The first two modes are connection-oriented, and the last two are connectionless:

- ***Reliable*** mode uses end-to-end acknowledgements from the destination to the source as the only way to *guarantee* delivery. This carries the penalty of requiring the end nodes to maintain state regarding each packet in flight over the entire E2E path, which can be substantial in high bandwidth-×-delay product environments. Reliable mode has the option of using FEC to reduce the probability of packet retransmission due to bit corruption.

- ***Near-reliable*** mode is highly reliable, but does not *guarantee* delivery, instead using the custody transfer [130] approach, which splits the ACK loop at intermediate realms at the cost of buffering ResTP segments in each PoMo gateway until acknowledged by the next realm along the path. Since the gateway uses split ARQ and immediately returns TCP ACKs to the source, the assumption is that ResTPs reliable ARQ-based delivery will succeed using SNACKs (selective negative acknowledgements) [81] supplemented by a limited number of (positive) ACKs. This can be more bandwidth-efficient than full source–destination reliability. However, the possibility exists of confirming delivery of data that the gateway cannot actually deliver to its final destination.

- ***Quasi-reliable*** mode uses only open-loop error recovery mechanisms such as FEC and erasure coding across multiple paths if available [239], thus eliminating ACKs and ARQ entirely. In this mode the strength of the coding can be tuned using cross-layer optimizations based on the quality of the channel being traversed, available bandwidth, and the application's sensitivity to data loss. This mode provides an arbitrary level of statistical reliability but without absolute delivery guarantees.

- ***Unreliable*** mode relies exclusively on the FEC of the link layer to preserve data integrity and does not use any error correction mechanism at the transport layer. Cross-layering is used to vary the link FEC strength.

The multipath mechanism may be useful in implementing any of the *reliability modes*, depending on the *service type* selected, and the graph of the underlying topology.

4.1.2 Mechanism Tradeoffs

In designing a transport protocol with multiple reliability modes we are seeking to understand the mechanism tradeoffs, and balance the contribution of each mechanism with the associated costs. These costs may be in the form of basic resources such as processing, memory, and bandwidth. They may also be more subtle, such as increased protocol complexity, security vulnerabilities, or reduced confidence of accurate delivery. Separating the available mechanisms into distinct modes is useful for implementing and comparing them. Future work will automate the mechanism selection process so that instead of pre-selecting a mode, the transport protocol can select and tune mechanisms on-the-fly in order to adapt to changing path conditions.

4.1.3 ResTP Multipath Simulations

There are many mechanisms that may be used to increase resilience. In these simulations we are specifically interested in the use of multiple end-to-end (E2E) paths through path diversification. We have simulated ResTP assuming the presence of the PostModern Internetwork Architecture (PoMo) [24], which provides cross-layer information from lower layers with which to make path selection decisions.

Simulation Setup

Using the ns-2 simulator[1] [240] we have implemented the multipath mechanism of ResTP and compared its performance to traditional single-path data transmission. Each source

[1]This early work was performed before the ns-3 simulator was available

sends n redundant (identical) data packets over the n paths it has requested. Because the backup paths are *already in use* there is no loss of data in the event of a failure unless *all* n paths are compromised. In order to limit the computational resources required to run the simulations, the data rates and bandwidths were scaled down from their actual values by two orders of magnitude. Each data point was averaged over 100 runs to eliminate aberrations caused by randomness in the simulation. The plots presented in this section represent a combined total of 17,600 simulation runs.

Fault-Tolerant Topologies

It is intuitively obvious that using a multipath transport protocol will not yield any benefit in terms of resilience unless multiple logical and physical paths are present. For this reason we have confined ourselves to simulating on topologies which are bi-connected or better.

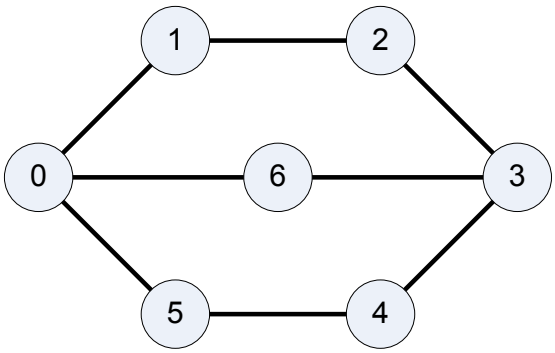

Figure 4.2: Synthetic seven-realm interconnection topology

Synthetic Topology The first topology we are considering is synthetic and created in order to characterize the behavior of the multipath mechanism on a small scale. It consists of seven distinct realms, interconnected as shown in Figure 4.2. Each of the links has a bandwidth of 1 Mbps and a propagation delay of 10 ms. Each of the realms has a randomly generated (using BRITE [241]) well-connected internal network of 15–20 nodes

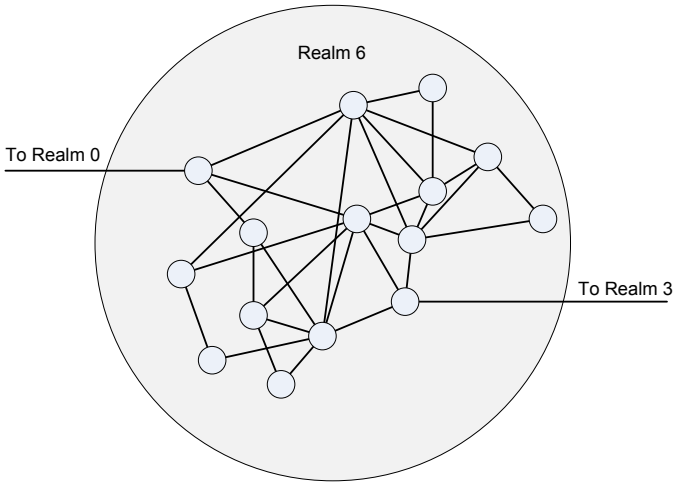

Figure 4.3: Internal topology of realm six

interconnected by 1 Mbps links. Figure 4.3 shows the internal network of realm 6 as an example.

ISP Backbone Topology To further characterize the multipath behavior in a more connected environment we used a map of AT&T's backbone network, Figure 4.4. This was obtained from the Rocketfuel project [222], which uses a probing technique to determine physical topology. We removed links which were deemed improbable to exist on routes geographically distinct from other existing fiber paths, for example a direct connection between San Francisco, CA and Dallas, TX is unlikely to exist at the physical layer, and was removed. We also removed any stub-nodes since they could not be part of any end-to-end disjoint paths. The 25 nodes in the topology are interconnected with 1 Mbps links for the purpose of creating a tractable simulation.

Traffic Patterns

On the synthetic topology, two nodes from each realm are randomly selected as traffic sources, each with a randomly selected destination in a different realm (no destination

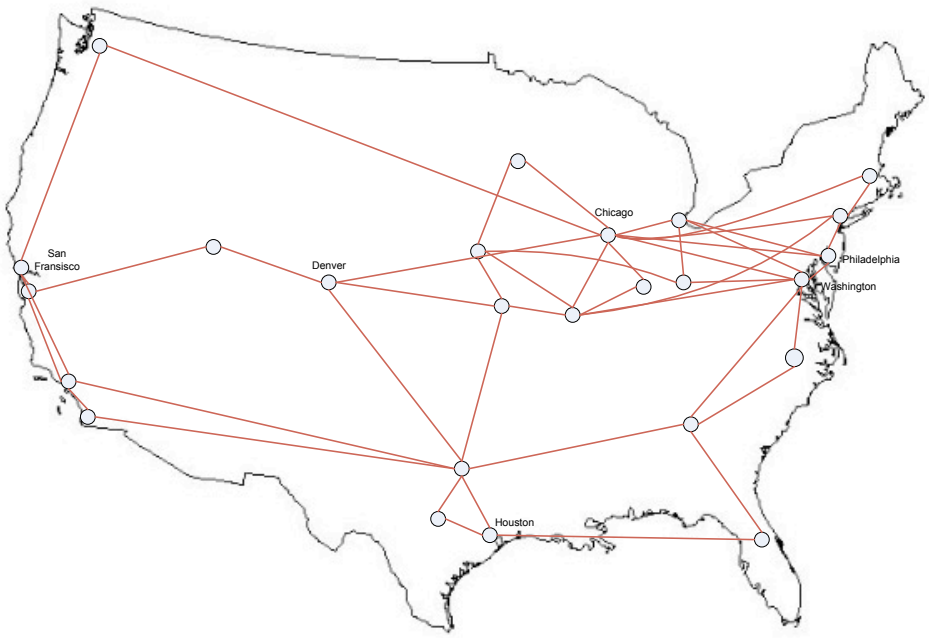

Figure 4.4: simplified AT&T backbone topology

node occupied the same realm as its respective source). In each of the failure scenarios, the application sending data-rate is set to 50 Kb/s so that congestion is not a source of loss. For each of the the load scenarios, the application sending data-rate is varied between 50 and 500 Kb/s to observe the effect of multipath on congestion. Each path is selected at the realm level, with traffic traversing the realm via one randomly-selected node and the shortest available path.

For the ISP topology (Figure 4.4), we are concerned with performance in the presence of a greater number of available diverse paths. For this topology we defined (source, destination) pairs as follows: (Chicago, Houston), (Washington, San Francisco), and (Philadelphia, Denver). These are selected because each pair has a minimum of 3 link-disjoint paths between them. All the ISP scenarios are run with an application sending data-rate of 50 Kb/s. The paths for this scenario specify each transit node.

For each scenario, we compare the performance to a baseline flow routed using a traditional single-path algorithm. This is the curve labeled “sp” on the graphs. The other

three curves show the performance of ResTP using the path selection algorithm described in Section 3.2.4 while requesting n diverse paths. It should be noted that the $n = 1$ case does not carry any implication of improved reliability over the sp case, it is only included here for completeness and to show that our algorithm does not cause any degradation in performance. The simulation was run with a 20 second warm-up time after traffic began, at which time the link failure scenario was applied. Traffic was sent for an additional 480 seconds, and all queued traffic was allowed to propagate to the destination before the simulation terminated.

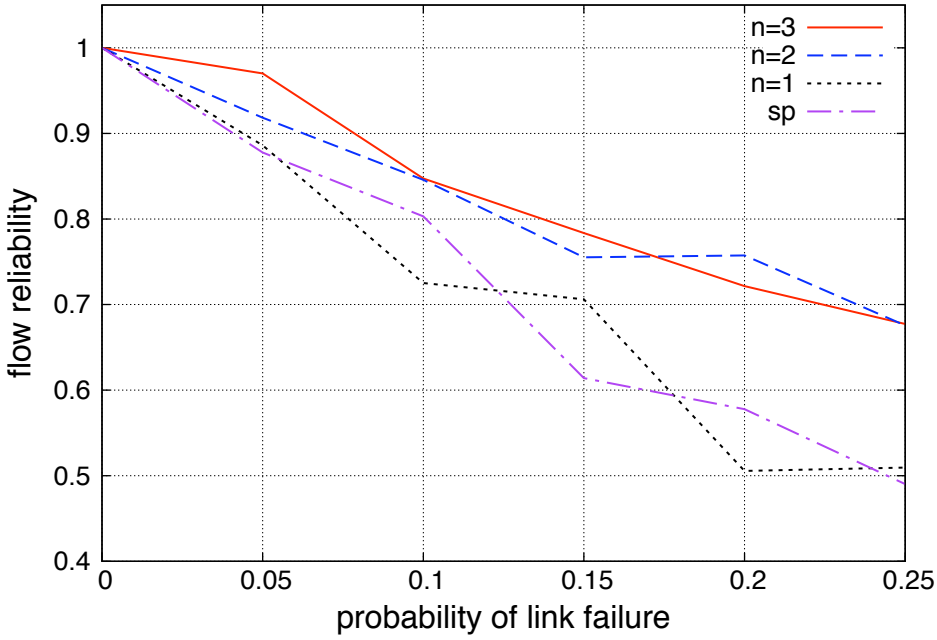

Figure 4.5: Comparison of flow reliability for synthetic topology

Flow Reliability with Link Failures

The link failure scenario is defined as follows. After the simulation warm-up period, each link failed with a uniform independent probability. This probability is varied between 0 and .25 to evaluate performance under varying severities of failure.

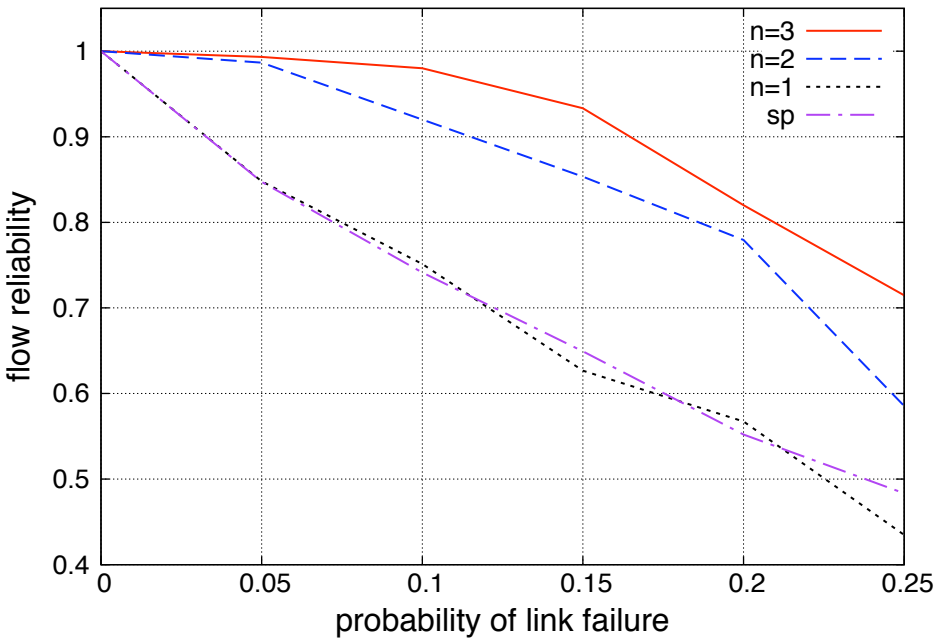

Figure 4.6: Comparison of flow reliability for AT&T topology

Our primary metric for evaluating the performance of multipath is *flow reliability*. This is shown as the fraction of flows which continue delivering data during a link failure scenario. On the synthetic topology we observe a maximum reliability improvement of 20% over single-path routing, shown in Figure 4.5. We also note that the performance of multipath with $n = 2$ and $n = 3$ is nearly identical. We attribute this to the low degree of connectivity in the network, which results in fewer high-diversity paths being available to ResTP. On the ISP topology we observe a maximum improvement of nearly 30% over single-path routing, shown in Figure 4.6. In this case the performance of multipath with $n = 3$ is consistently 5–10% better than the performance with $n = 2$.

Performance with respect to Load Clearly there is a cost to increasing reliability with this mechanism, and the tradeoff we are making in increased traffic in the network, and eventually increased congestion. We use the 7-realm topology to evaluate the effects of increased load. Figure 4.7 shows the decrease in performance due to congestion losses

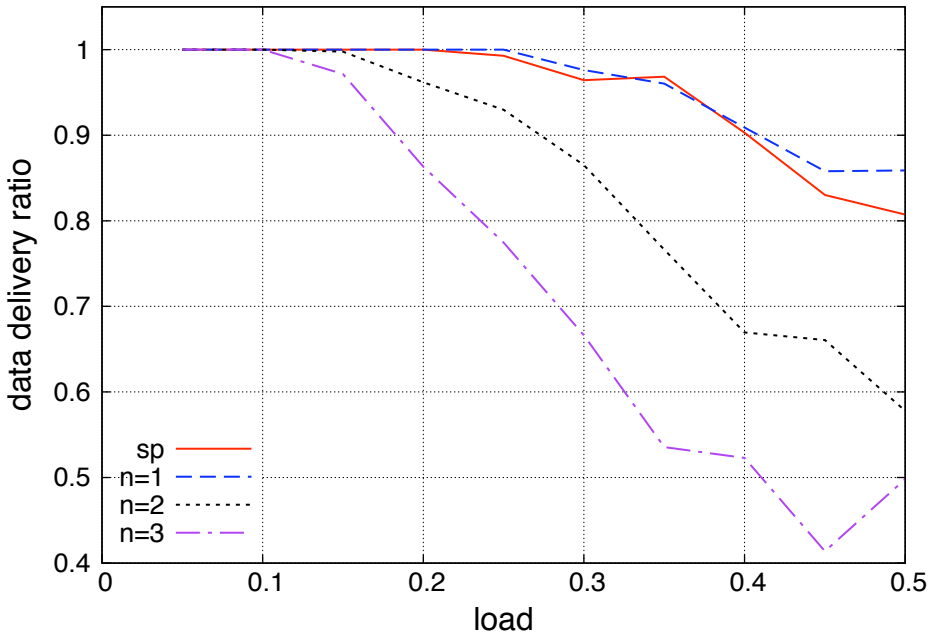

Figure 4.7: Reduction in performance due to congestion as traffic load increases

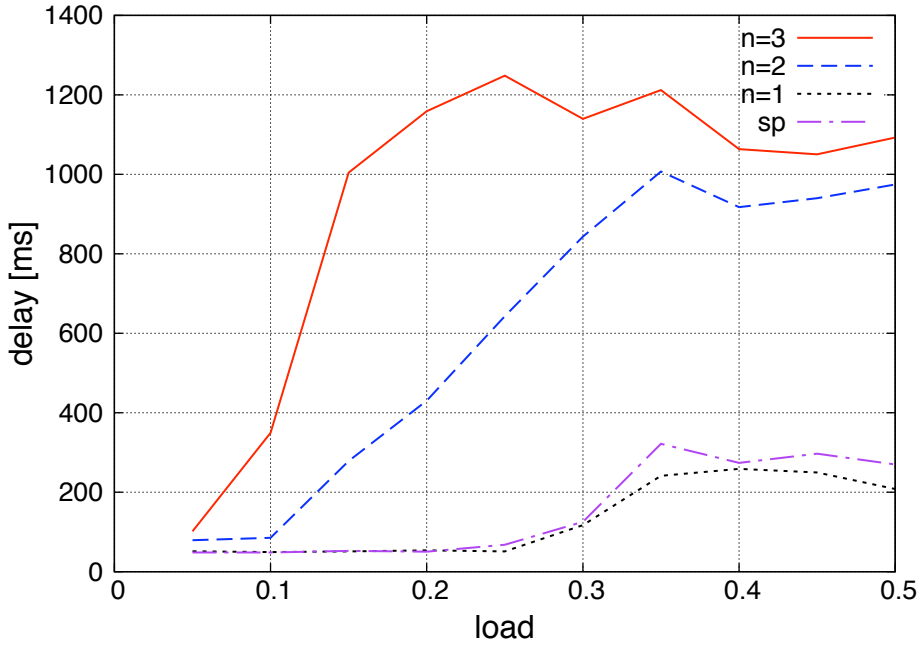

Figure 4.8: Increase in delay as traffic load increases

as the load increases in the network. Figure 4.8 shows the corresponding increase in end-to-end delay caused by queuing in the network as congestion increases. Again these results are topology dependent, in that a more highly connected graph will have the additional load of multipath spread across a greater number of alternate links.

4.1.4 Summary

In this section we introduced the ResTP protocol architecture, assuming the presence of an internetwork layer such as PoMo to provide cross-layer information with which to make intelligent path selection decisions. We have shown a 20–30% performance improvement in the presence of link failures when diversity is available in the underlying network graph. In the next section we will describe AeroTP, a domain-specific implementation of the ResTP architecture.

4.2 Airborne Telemetry Transport

One challenged environment which we have spent a significant level of effort analyzing are highly-dynamic airborne telemetry networks. Due to the end-to-end challenges introduced by the very high mobility of the scenario it provides an excellent environment for a case study on the benefits of the ResTP protocol. The TmNS (telemetry network system) environment consists of three realms, the first being a wired IP network internal to each airborne node, the second being the highly dynamic MANET connecting the airborne nodes to the ground station, and the third being the wired IP network on the ground. Due to the fact that the two IP networks are not PoMo-enabled, we have developed a domain-specific version of ResTP called AeroTP that operates edge-to-edge within the wireless MANET, and is translated at the borders of this realm to TCP or UDP using gateways [235]. This section gives an overview of the environment, as well as the protocol suit we have developed to achieve higher performance in the face of these challenges than was previously possible.

4.2.1 Overview of the ANTP Suite

Highly dynamic airborne tactical networks pose unique challenges to end-to-end data transmission. The current TCP/IP-based Internet architecture is not designed to function in this environment, however this architecture is almost exclusively used within the embedded components that make up modern tactical communications systems, as well as across the Global Information Grid (GIG) [242]. This necessitates that any domain-specific solution designed to optimize performance in a tactical environment must at the same time maintain some compatibility with the TCP/IP stack. This section presents the design and evaluation of a protocol suite that is optimized for the tactical environment, while maintaining edge-to-edge compatibility with the legacy Internet architecture.

These protocols include: *AeroTP* – a TCP-friendly transport protocol introduced in [231] and further evaluated in [234, 236], with multiple reliability and QoS modes that is an implementation subset of ResTP (described in Section 4.1), *AeroNP* – an IP-compatible network protocol (addressing and forwarding) introduced in [243], and *AeroRP* – a routing protocol introduced in [243] and further evaluated in [232, 237], which exploits location information to mitigate the short contact times of high-velocity airborne nodes. This protocol suite is designed to perform well in an environment in which rapidly-changing topology prevents global routing convergence, as well as those in which long-lasting stable end-to-end paths do not exist.

While these protocols are designed to perform well in a broad range of highly-dynamic scenarios, the airborne test and evaluation community in particular has recognized the need to replace an aging telemetry communication architecture with a full multihop network protocol suite such as the one described here. Traditionally, telemetry communication has consisted primarily of point-to-point links from multiple sources to a single sink. More recently, with the increasing number of sources in the typical telemetry test scenario, there is a need to move to networked systems in order to meet the demands of bandwidth and connectivity. This need has been recognized by various groups, including the Integrated Network Enhanced Telemetry (iNET) program for Major Range and Test Facility Bases (MRTFB) across United States [244]. The current TCP/IP-based Internet architecture is not designed to address the needs of telemetry applications [245] and there remain a number of issues to be solved at the network and transport layers [246]. In particular, given the constraints and requirements of the aeronautical environment, the current Internet protocols are not suitable in a number of respects. These constraints include the physical network characteristics such as topology and mobility that present severe challenges to reliable end-to-end communication. In order to build a resilient [8] network infrastructure, we need cross-layer enabled protocols at the transport, network,

and MAC layers that are particularly suited for airborne networks. At the same time, there is a need to be compatible with both TCP/IP-based devices located on the airborne nodes as well as with ground-based control applications. Therefore, the new protocol suite must be fully interoperable with TCP/UDP/IP via gateways at the telemetry network edges. Due to the limited bandwidth in telemetry networks and a priori knowledge of communication needs of a given test, the iNET community is developing a TDMA (time division multiple access)-based MAC for this particular environment [247]. We will revisit the telemetry-range case study later in the chapter to illustrate several features of our protocol suite.

It is important to note that while tactical networks constrain some aspects of network operations, there are also aspects that can be *exploited* by domain specific protocols, such as the knowledge of the airborne node location and trajectory. Previous research has developed several intelligent network protocols in the context of mobile ad hoc networks (MANETs) and wireless sensor networks (WSNs) that attempt to exploit additional information available [248, 249]. However, in order to achieve this, we need to facilitate cross-layering across the multiple layers. For example, location and trajectory information can be used to find better paths if there exists a mechanism, either implicit or explicit, for information exchange between the physical and network layers. As generally recognized, strict layering in the network stack is not particularly suitable for wireless networks due to mobility, limited bandwidth, low energy, and QoS requirements. Therefore, it is commonly agreed upon that a tighter, more explicit, yet careful integration amongst the layers will improve the overall wireless network performance in general; and in the case of highly-dynamic, bandwidth-constrained networks may provide the only feasible solution that meets the requirements of tactical applications.

AeroTP is a transport protocol designed with several reliability modes to address the requirements of different traffic classes based on ResTP. This relies on the new network

protocol *AeroNP*, which is fundamental to the architecture because it enables explicit cross-layer interactions between layers by passing congestion, QoS, and packet corruption information up and down the protocol stack. Furthermore, its header carries node and device identifiers, along with location and trajectory information that is critical for the routing protocol. Lastly, we define a location-aware, highly-adaptive routing algorithm *AeroRP* that utilizes the node location and trajectory to route packets through the telemetry network. Simulation results show that AeroRP significantly outperforms traditional MANET routing protocols that require an end-to-end path determined either proactively or on-demand. We introduce several new mechanisms to improve routing and forwarding efficiency in highly-dynamic networks. These include node discovery based on snooping, efficient directional-packet forwarding, and implicit congestion control.

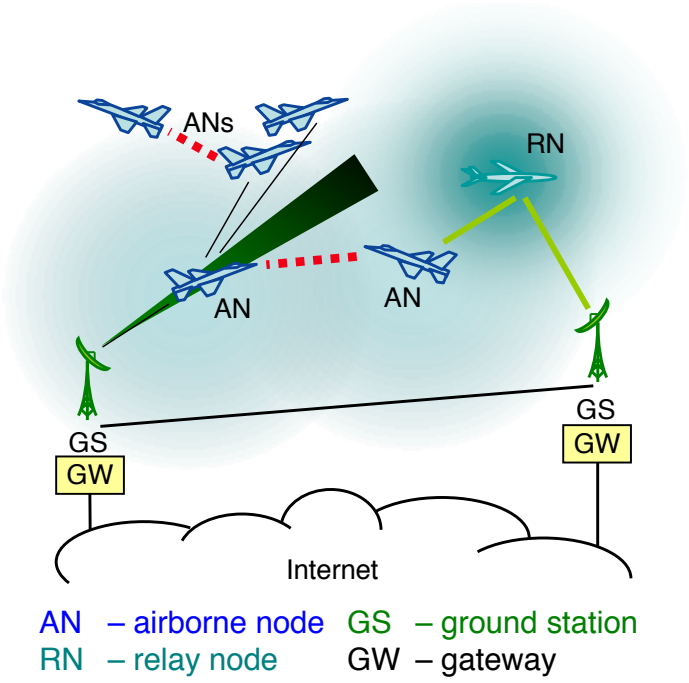

Figure 4.9: Dynamic airborne tactical environment

4.2.2 Networking Challenges in Airborne Tactical Networks

A typical airborne tactical network as depicted in Figure 4.9 consists of three types of nodes: airborne nodes (AN)[2], ground stations (GS) and relay nodes (RN). The airborne nodes (e.g. reconnaissance and combat aircraft, piloted or unpiloted) contain a variety of data collection devices that are primarily IP devices such as cameras, hereafter referred to as *peripherals*. ANs house omnidirectional antennas with relatively short transmission range. The GSs are located on the ground (stationary or portable) and typically have a much higher transmission range than that of an AN through the use of large steerable antennas. In point-to-point communication mode, the GS tracks a given AN across some geographical space. However, due to the narrow beam width of the antenna, a GS can only track one AN at a time. The GS also houses a gateway (GW) that connects the airborne network to the Internet and several terminals that may run control applications for the various devices on the AN. Furthermore, the GSs can be interconnected to do soft-handoffs from one to another while tracking an AN. The RNs are dedicated airborne nodes to improve the connectivity of the network. These nodes have enhanced communication resources needed to forward data from multiple ANs and can be arbitrarily placed in the network. There are a number of challenges to communication protocols in this environment:

- *Mobility*: The airborne nodes can travel at speeds as high as Mach 3.5 (1191 m/s), possibly faster in the future; the extreme is then two ANs closing with a relative velocity of Mach 7 (2382 m/s). Because of high speeds, the network is highly dynamic with constantly changing topology.

[2]These are also referred to as test articles (TA) in our publications targeted specifically to the telemetry environment

- *Constrained bandwidth*: Due to the limited spectrum allocated to tactical networks, and the high volume of data to be transferred, particularly for situational awareness, the network is severely bandwidth-constrained.

- *Limited transmission range*: The energy available for data transmission on some ANs is limited due to power and weight constraints, particularly with smaller vehicles, requiring multihop transmission from AN to GS.

- *Intermittent connectivity*: Given the transmission range of the AN and high mobility, the contact duration between any two nodes may be extremely short leading to network partitioning. Furthermore, the wireless channels are subject to interference and jamming.

In Table 4.1, we use the numerical baseline values from the network characteristics of the iNET telemetry network [245] case to estimate the expected stability of the links in such a network. Even with optimistic transmission range, the contact duration between two neighboring nodes can be as low as 15 s. Note that in a multihop scenario with lower transmission power, the contact duration between an airborne node and ground station can be even shorter. At the same time there is no maximum contact duration, so a protocol used in this environment will need to be able to make efficient use of available spectrum and manage multiple traffic priorities for long-lived flows as well as short-lived ones.

4.2.3 Existing Protocols & Architectures

Given that the tactical communications community relies in large part on existing TCP/IP-based embedded devices and communicating the data to existing IP-based applications, it is important to understand the implications of using the traditional Internet protocols

Table 4.1: Link stability analysis

Scenario	**Tx range [km]**	**Relative velocity**	**Contact duration [s]**
Single hop best case			
AN – GS	260	206 m/s	2520
AN – AN	28	412 m/s	135
Single hop worst case			
AN – GS	185	1191 m/s Mach 3.5	300
AN – AN	18	2382 m/s Mach 7.0	15

(UDP, RTP, TCP, and IP) in a highly dynamic environment. There has also been substantial research in transport protocols specific to satellite networks and routing protocols for MANETs, both of which share some characteristics with airborne tactical networks. This section briefly mentions some drawback of traditional protocols for the airborne environment, in addition to those mentioned in Chapter 2.

Transmission Control and User Datagram Protocols

In the tactical airborne environment we expect to have multiple classes of traffic with different characteristics, different tolerance of loss, and different priorities. Neither TCP or UDP have the capability to express differentiated levels of precedence or QoS to permit the network to meet these requirements. A number of these shortcomings have been researched, and a few alternative protocols exist, such as SCPS-TP (Space Communications Protocol Standards – transport protocol) [81] described in Section 2.4, from which we can draw some mechanisms but are only a partial solution.

SCPS-TP

While SCPS-TP solves a number of the problems associated with airborne tactical networks, and our solution uses some of the same mechanisms, SCPS-TP is not ideal for our application because it relies too heavily on channel condition information which is either pre-configured or learned gradually over multiple end-to-end connections. This process cannot adapt adequately to the rapidly changing airborne environment, or opportunistically make use of available bandwidth on a hop-by-hop basis.

Internet Protocol (IP)

The traditional wired Internet uses IP at the network layer, with various routing protocols such as OSPF [250], RIP [251], and BGP [252]. TCP over IP adds a header of 40 bytes per packet. This overhead becomes significant if there are many small packets (e.g. control traffic), which is the case with the per-segment acknowledgements of TCP. The current Internet architecture is based on the fundamental assumption of long-lasting, stable links that does not hold true for a Mach-speed airborne network, which not only challenges TCP as described above, but also network routing. Internet protocols require convergence of the routes and do not natively support dynamic topologies inherent in the airborne telemetry environment. IP also does not efficiently support the inherent 2-level hierarchy caused by each AN containing a limited number of individually-addressed peripherals. Furthermore, the current architecture was not systematically designed to be a distributed solution to a global optimization problem [253] and does not support explicit cross-layer information exchange to leverage unique information available in the network such as position and trajectory.

Ad Hoc Routing Protocols

In order to support MANETs (mobile ad hoc wireless networks), several routing protocols have been developed that adapt to changes in topology. Reactive routing protocols such as AODV [254] and DSR [255] attempt to construct source-to-destination paths on demand and are not suitable because of the delay involved in finding paths and because such paths may not be valid for long enough in a highly-dynamic network. On the other hand, proactive routing protocols such as DSDV [256] and OLSR [257] forward packets on a hop-by-hop basis and depend on route convergence. This generates excessive overhead due to frequent route updates (assuming convergence is even possible) and is not suitable for a bandwidth-constrained airborne network.

There are several other protocols that adapt to mobility by forwarding packets one hop at a time without attempting to construct the entire path. These include simplistic approaches such as flooding and other greedy algorithms that send multiple copies in the network [258]. More complex routing schemes leverage specific information from the network. Most notable are the location-based routing protocols such as LAR, DREAM, SIFT, and GRID, APRAM, and anticipatory routing [259–265] that use GPS coordinates of the nodes to determine the next hop. Aeronautical telemetry networks require self-organizing protocols like the one proposed in [266], but designed for high relative node mobility.

Furthermore, airborne tactical networks require the routing protocol to be highly adaptive based on the particular mission requirements. Most existing routing mechanisms are *unimodal*, wherein the algorithm is optimized for a specific mode of operation. A varying set of operating conditions and service requirements justify the need for a domain-specific *multimodal* protocol that inherently supports multiple modes of operation.

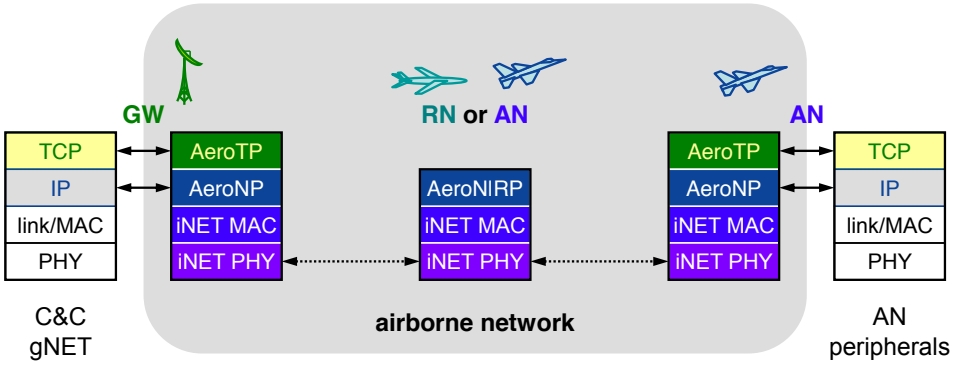

Figure 4.10: Airborne network protocol architecture

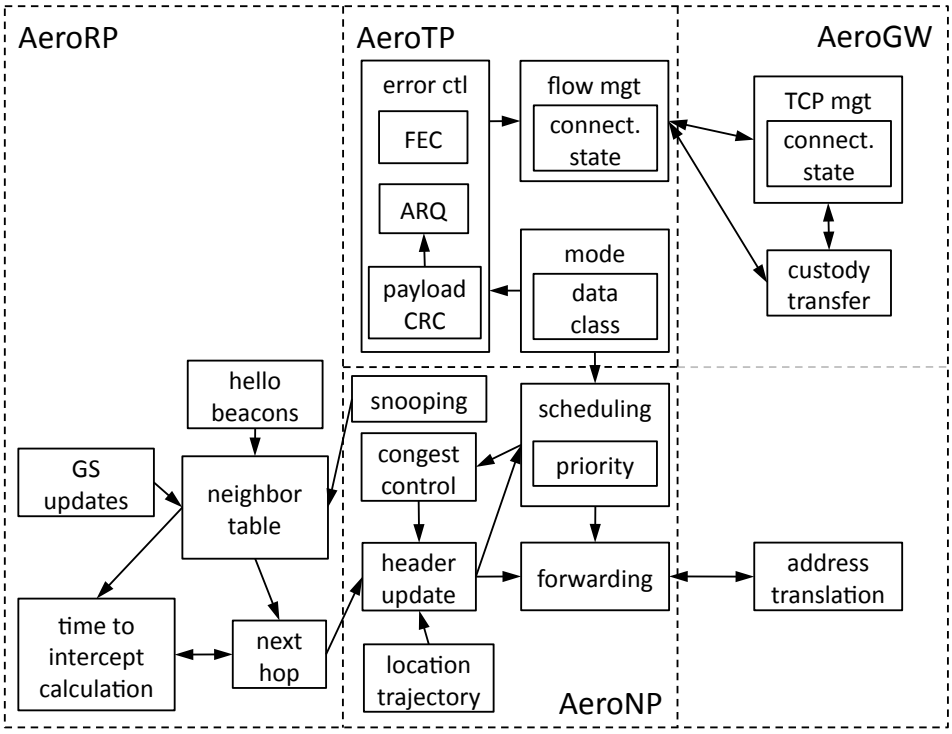

Figure 4.11: Airborne network protocol functional block diagram

4.2.4 System Architecture

This section describes a new set of protocols designed for the aeronautical environment: AeroTP, a TCP-friendly[3] transport protocol; AeroNP, an IP-compatible network protocol; and the AeroRP routing protocol for highly-dynamic airborne nodes. The major functions of each of these protocols, as well as the control-plane relationships between them are shown in Figure 4.11. The communications we are concerned with can include any type of packetized information and may be directed from GS to AN, from AN to GS, or from AN to AN, and may use an intermediate RN if available. As mentioned in Section 4.2.3, both the source and destination for data transmitted may be native Aero-protocol devices or TCP/IP-based systems, however the IP protocol stack is not suitable for use within the airborne network itself. To overcome this challenge without requiring a total redesign of all sensors, peripherals, applications, and workstations, we introduce the Aero Gateway (AeroGW) [268]. The gateway concept, often referred to as *protocol boosters* [269] or *performance enhancing proxies* [270], is well established as a mechanism for bridging between disparate network environments. In this case its operation is similar to TCP-Splice [271], however instead of splicing TCP with TCP, it will translate TCP (and UDP/RTP) to AeroTP and IP to AeroNP. This functionality resides in the AeroGW, which is incorporated into each ground station and airborne node. An expected use case is shown in Figure 4.10 with a ground station and airborne node communicating using standard TCP/IP protocol stacks, which are translated to AeroTP/NP for greater dependability [272] and performance in the wireless network. Due to the requirement for backward compatibility we expect this use case to be the most common, however there is no limitation in our design preventing nodes from running the AeroTP/NP stack natively and bypassing the gateway. Nodes running the AeroTP/NP stack natively, with

[3]Note that we use the term "TCP-friendly" in a more general sense than the established term "TCP-friendly rate control" (TFRC) [267]

gateways performing custody transfer at the edges of the wireless network is directly analogous to running ResTP end-to-end with PoMo gateways at realm boundaries.

Table 4.2: Feature Comparison of AeroTP, TP++, UDP, and TCP Variants

Feature	**AeroTP (ResTP)**	**TP++**	**UDP**	**TCP NewReno**	**(CU)BIC -TCP**	**T/TCP**	**SCPS-TP**
TCP Compatible	friendly	no	no	yes	yes	yes	interop
UDP Compatible	friendly	no	yes	no	no	no	no
3-way handshake	no	no	no	per-flow	per-flow	per-endpoint	per-endpoint
partial-path support	yes	no	yes	no	no	no	no
header integrity check	CRC-16	chksum	no	no	no	no	no
data integrity check	CRC-32	chksum	16-bit chksum	16-bit chksum	16-bit chksum	16-bit chksum	16-bit chksum
error correction	variable FEC	FEC	no	no	no	no	no
aggregated ACKs	yes	yes	no	optional	optional	no	yes
selective repeat	yes	yes	no	optional	optional	no	yes
negative ACKs	optional	no	no	no	no	no	optional
multipath friendly	yes	yes	no	no	no	no	no
flow control	x-layer	out-of-band signals	no	windowed	windowed	windowed	windowed
congestion ctrl	x-layer AeroNP backpressure	none	none	slow-start, AIMD, fast rexmit	slow-start, (CU)BIC, fast rexmit	estimate, AIMD	estimate, Vegas, fast rexmit
error control	hybrid, modular, adaptive,	hybrid, modular	none	ARQ	ARQ	ARQ	ARQ
reliability modes	reliable nearly-reliable quasi-reliable best-effort	reliable quasi-reliable	best-effort	reliable	reliable	reliable	reliable

4.2.5 AeroTP: TCP-Friendly Transport Protocol

AeroTP is a new domain-specific transport protocol designed to meet the needs of the highly-dynamic network environment while being *TCP-friendly* to allow efficient splicing with conventional TCP at the AeroGWs in the GS and on the AN. Thus it transports TCP and UDP through the tactical network, but in an efficient manner that meets the needs of this environment: disruption tolerance, dynamic resource sharing, QoS support for fairness and precedence, real-time data service, and bidirectional communication. Table 4.2 identifies a number of key features of AeroTP and compares it to other modern and traditional transport protocols. AeroTP has several operational modes that support

different service classes: reliable, nearly-reliable, quasi-reliable, best-effort connections, and best-effort datagrams. The first of these is fully TCP compatible, the last fully UDP compatible, and the others TCP-friendly with reliability semantics matching the needs of the mission and capabilities of the airborne network. The AeroTP header is designed to permit efficient translation between TCP/UDP and AeroTP at the gateway as described in Section 4.2.5.

AeroTP performs end-to-end data transfer between the edges of the airborne network and either terminates at native Aero devices or splices to TCP/UDP flows at the AeroGWs. Transport-layer functions that must be performed by AeroTP include connection setup and management, transmission control, and error control, shown in Figure 4.11.

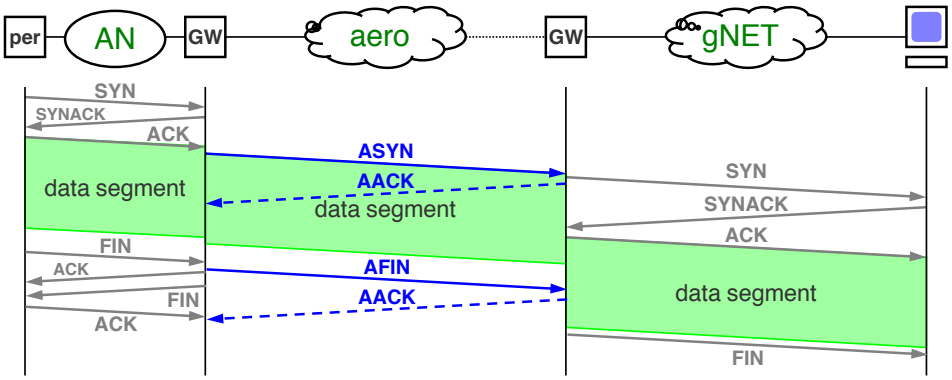

Figure 4.12: AeroTP connection setup

Connection Management and Transmission Control

AeroTP uses connection management paradigms suited to the wireless network environment. An alternative to the overhead of the three-way handshake is an opportunistic connection establishment in which data can begin to flow with the ASYN (AeroSYN) setup message (shown in Figure 4.12). The flow of data is originated by a peripheral sensor (per) as a standard TCP session, translated into an AeroTP session by the gateway to traverse the airborne network, and then translated back into a standard TCP session

by the gateway on the ground. The TPDU (transport protocol data unit) size may be discovered using the standard path MTU discovery mechanism [97], however given the specialized nature of these networks it is expected that the best performance will be achieved by setting the peripherals to use an appropriate MTU as determined by the slot size of the underlying TDMA MAC [247]. Closed-loop window-based flow and congestion control with slow start is not appropriate to the highly-dynamic nature of this network, therefore we use an open-loop *rate-based* transmission control with instrumentation from the network layer and determine an initial rate, with backpressure to control congestion, as described in Section 4.2.7 for AeroNP. Error control is fully decoupled from rate control [57, 68], and is service specific as described below.

Segment Structure and Gateway Functionality

AeroTP is *TCP-friendly*, meaning it is designed to efficiently interoperate with TCP and UDP at the gateways. To support this, AeroGW functionality [273, 274] provides IP–AeroNP translation [243] and TCP/UDP–AeroTP splicing. A packet may pass through two gateways on its path from source to destination. The ingress gateway converts the TCP segments to AeroTPDUs, while the egress gateway converts AeroTPDUs to TCP segments. It should be noted that ingress and egress gateways are not additional network elements in the tactical environment, but rather the gateway functionality will be built into ANs and GSs.

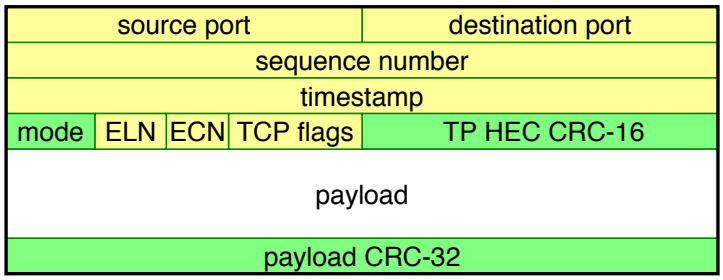

Figure 4.13: AeroTP TPDU structure

The AeroTPDU is shown in Figure 4.13. Since bandwidth efficiency is critical, AeroTP does not encapsulate the entire TCP/UDP and IP headers, but rather the gateway converts between TCP/UDP and AeroTP headers. Some fields that are not needed for AeroTP operation but are needed for proper end-to-end semantics are passed through, such as the source and destination *port number*, *TCP flags*, and the *timestamp*. The *sequence number* allows reordering of packets due to erasure coding (as with TP++ [275]) over multiple paths or AN mobility, and is either the TCP byte-sequence number or a segment number, depending on the AeroTP transfer mode described below. The *HEC* (header error check) field is a strong CRC (cyclic redundancy check) on the integrity of the header to detect bit errors in the wireless channel. This allows the packet to be correctly delivered to AeroTP at the destination where a corrupted payload can be corrected on an end-to-end basis using FEC (forward error correction). A *payload CRC* protects the integrity of the data edge-to-edge across the airborne network in the absence of a separate AeroNP or link layer frame CRC, and enables measurement of the bit-error-rate for error-correction code adaptation depending on the transfer mode. This method of error detection and correction implies that AeroNP does not necessarily drop corrupted packets at intermediate hops, which is a key difference from IP forwarding semantics [86, 144].

Error Control and QoS-Based Transfer Modes

Based on the application requirements, there will be a number a classes of data being transmitted over the tactical network. For this reason, AeroTP supports multiple transfer modes that are mapped to different traffic classes: reliable connection, near-reliable connection, quasi-reliable connection, unreliable connection, and unreliable datagram. These modes are slightly modified from the set found in ResTP to support the specific needs of the telemetry applications and environment.

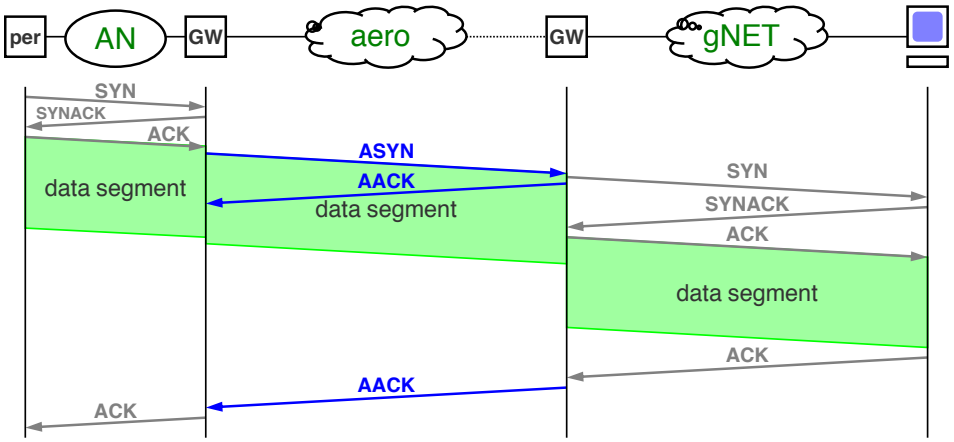

Figure 4.14: AeroTP reliable connection transfer mode

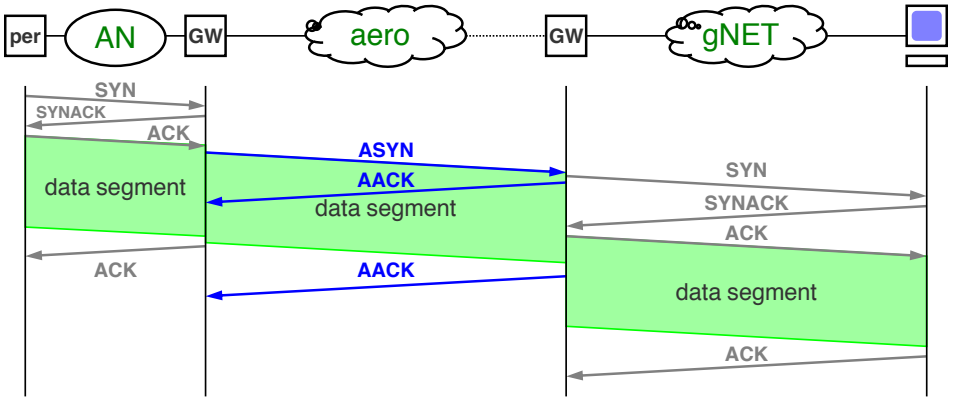

Figure 4.15: AeroTP near-reliable transfer mode

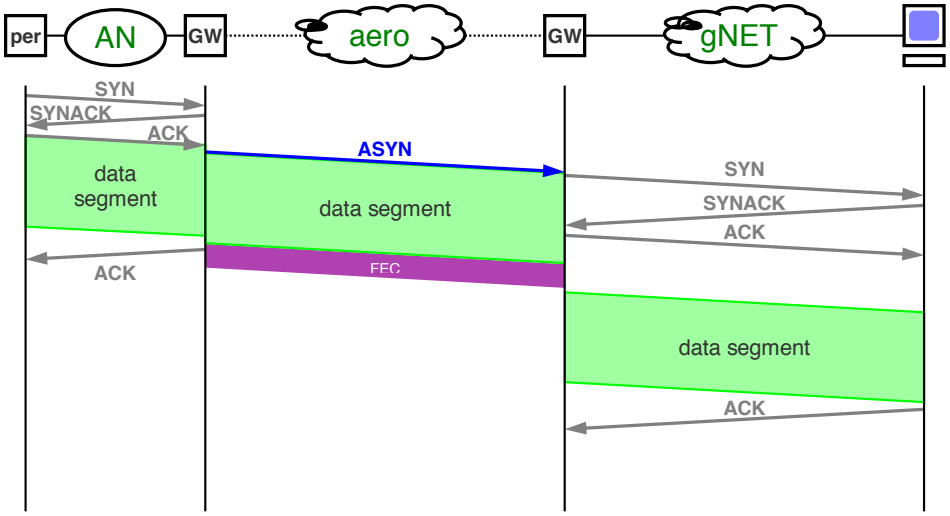

Figure 4.16: AeroTP quasi-reliable transfer mode

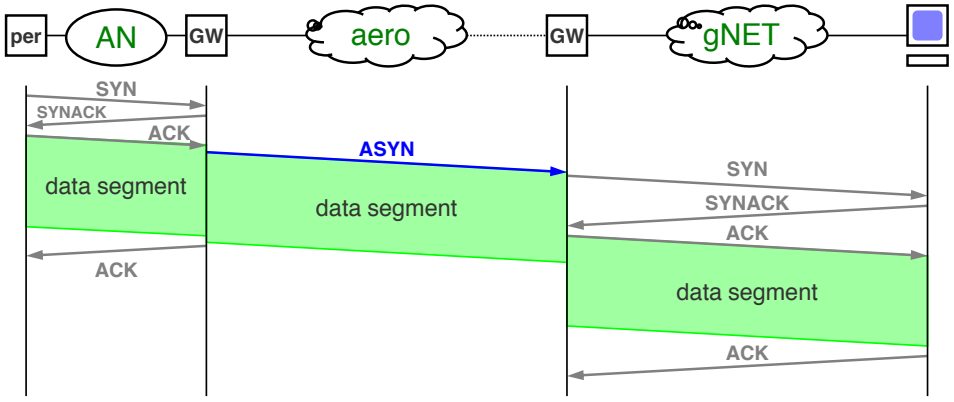

Figure 4.17: AeroTP unreliable connection transfer mode

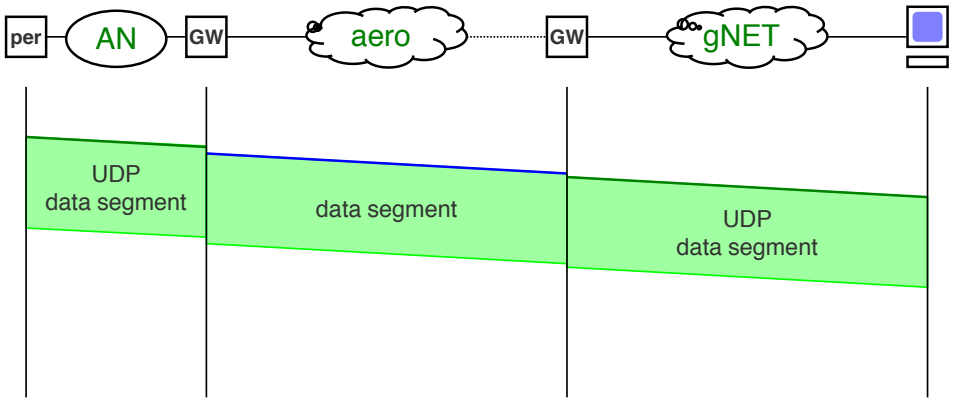

Figure 4.18: AeroTP unreliable datagram transfer mode

All modes except unreliable datagram are connection-oriented for TCP-friendliness and use sequence numbers so that packets may follow varying or multiple paths and be reordered at the AeroTP receiver.

- ***Reliable connection*** mode (Figure 4.14) must preserve end-to-end acknowledgement semantics from source to destination as the only way to *guarantee* delivery. We do this using TCP ACK passthrough, which has the disadvantage of imposing TCP window and ACK timing onto the AeroTP realm, but will never falsely inform the source of successful delivery.

- ***Near-reliable connection*** mode (Figure 4.15) uses a custody transfer mechanism similar to that used in DTNs [129, 130, 276] to provide high reliability, but can not *guarantee* delivery since the gateway immediately returns TCP ACKs to the source on the assumption that AeroTPs reliable ARQ (automatic repeat request)-based delivery will succeed using SNACKs (selective negative acknowledgements) [81] supplemented by a limited number of (positive) ACKs as well as ELN (explicit loss notification) [144]. This still requires that the gateway buffer segments until acknowledged across the airborne network by AeroTP, but is more bandwidth-efficient than full source–destination reliability because TCP's ACK-clocked behavior only operates over the well-connected AN and ground-network (gNET) links, while allowing AeroTP to keep the assigned TDMA slots filled in the airborne network. However, the possibility exists of confirming delivery of data that the gateway cannot actually deliver to its final destination.

- ***Quasi-reliable connection*** mode (Figure 4.16) eliminates ACKs and ARQ entirely, using only open-loop error recovery mechanisms such as erasure coding, across multiple paths if available [239]. In this mode the strength of the coding can be tuned using cross-layer optimizations based on the quality of the wireless

channel being traversed, available bandwidth, and the sensitivity of the data to loss. This mode provides an arbitrary level of statistical reliability but without absolute delivery guarantees.

- ***Unreliable connection*** mode (Figure 4.17) relies exclusively on the link layer (FEC or ARQ) to preserve data integrity and does not use any error correction mechanism at the transport layer. Cross-layering may be used to vary the strength of the link-layer FEC.

- ***Unreliable datagram*** mode (Figure 4.18) is intended to transparently pass UDP traffic, and no AeroTP connection state is established at all.

4.2.6 Cross-Layer Mechanisms

Despite the fact that link-load aware routing was developed as a part of the first ARPANET routing protocol [277], cross-layered routing utilizing link and physical layer information in route selection is not widely used. The reason for this is twofold: firstly, intelligent cross-layer aware network protocols tend to be inherently complex, and secondly in wired networks, physical links are highly reliable and are frequently over-provisioned. This has led to shortest-path being the most widely deployed routing algorithm. It has been noted that this is clearly not sufficient for effective routing in wireless networks [278], which motivates the need to exploit available information through cross-layering to make better forwarding decisions at each node.

Table 4.3 shows knobs at each layer that enable higher layers to influence certain mechanisms at lower layers, based on the information made available through dials. For example, the transport layer influences path selection through the *forwarding mode* knob, thus requesting a certain level of reliability for given data flow.

Table 4.3: Knobs and dials for a telemetry network stack

Layer	Knobs	Dials	Influencing Layer
mission	policy & requirements	situational awareness	command & control
application	data class	status indicators	mission
transport	reliability mode	diversity, goodput, outage	application
network	ARQ, priority	paths, loss/errors	transport
link & MAC	ARQ & FEC settings	neighbors, BER	network
physical	coding scheme	location, SNR	link

The airborne protocols employ cross-layer optimizations not only among the transport (AeroTP) and network (AeroNP) protocols, but also with the MAC and PHY layer. This involves optimizing the tradeoffs in type and strength of FEC at the PHY layer with respect to channel conditions and BER (bit error rate), as well as optimizing TDMA parameters and slot assignment based on the transfer mode of AeroTP and QoS parameters (precedence and service type) of AeroNP. Additionally, the support for multicast and broadcast requires coordination of AeroNP routing with the broadcast capabilities of the MAC.

4.2.7 AeroNP: IP-Compatible Network Protocol

AeroNP is a network protocol designed specifically for the highly-dynamic airborne environment, however, given the IP-based end devices on the ground for command and control, as well as TCP/IP peripherals on the AN, it is critical for the airborne network protocol to be compatible with IP. The AeroGW converts IP packets to AeroNP packets and vice-versa. The key features of AeroNP are to provide explicit support for cross-layering messages discussed in Section 4.2.6, reduce overhead by providing an efficient addressing mapping from IP, and provide a strong header check to decode errored payloads that could be recovered by AeroTP error-correction mechanisms.

The AeroNP packet header format, shown in Figure 4.19, is 32-bits wide. The *version*

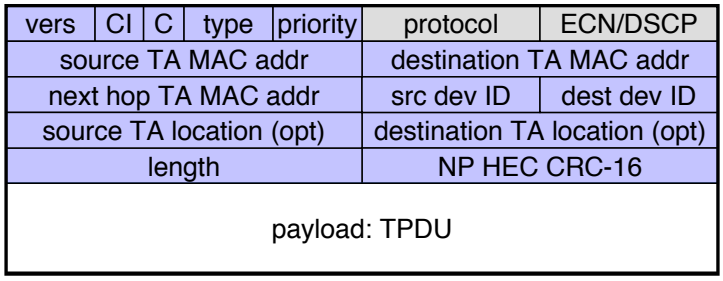

Figure 4.19: AeroNP packet structure

is the AeroNP protocol version, the congestion indicator (*CI*) is set by each node to notify the neighboring nodes of its congestion level as discussed later. The *type* and *priority* fields specify the QoS level of a given packet. The number of QoS classes can be customized for a given scenario. *Protocol* is the demux protocol identifier to which AeroNP hands off the packets. In order to provide IP transparency, the *ECN/DSCP* (explicit congestion notification/diffserv code point) nibble is carried over from the IP header. An AeroNP packet is inserted directly into a TDMA slot, and thus contains the *MAC addresses*: *source*, *destination*, and *next hop*. Significant efficiency can be gained if the AeroNP header does not carry the 32-bit source and destination IP addresses (or the even worse 128 bit addresses for IPv6). By performing an ARP-like address resolution process, the IP address can be mapped to MAC addresses in the AeroGW. However, each AN can have multiple peripherals, each of which has an IP address. Therefore, we include a *device-id* field in the header, and the $\langle$*MAC-address*, *device-id*$\rangle$ tuple is mapped to the peripheral IP address at the AeroGW. While dynamic mapping procedures are possible, it is more efficient to preload the translation table at the beginning of each mission. Optionally, *source* and *destination location* are included, which can be the GPS coordinates that are used in location-aware routing. The *length* indicates the actual length of the header in bytes. A strong check on the integrity of the header, *HEC*, is included to protect against bit errors. Unlike Internet protocols [86], the default behavior of AeroNP is to repair the corrupted bit and forward the errored packets to the transport

layer instead of dropping them at the network layer. The corruption indicator (C) bit is set by AeroNP to notify AeroTP that corruption has been experienced. This permits FEC at the transport layer to correct errors in the AeroTP quasi-reliable mode, as described in Section 4.2.5.

4.2.8 AeroRP: Location-Aware Adaptive Routing

The small contact duration among ANs results in frequent routing changes and is indicative of the need for an intelligent multihop routing protocol, supporting reliable communication over the highly dynamic physical topology. As discussed previously, existing routing mechanisms generate significant overhead and do not converge quickly (if ever) in the presence of frequent topology changes and hence are not suitable for highly-dynamic networks. The AeroRP routing protocol is specifically designed to address the issues related to highly mobile aeronautical environments. This protocol was designed and simulated in close collaboration with Abdul Jabbar. We utilize a number of mechanisms that have been researched independently for use in environments with characteristics similar to those of aeronautical telemetry:

- **Proactive behavior:** AeroRP is a fundamentally proactive routing protocol, but with limited updates thereby lowering protocol overheard.
- **Exploits cross-layer controls:** AeroRP is designed to exploit the explicit cross-layering support provided by AeroNP and the geographic node location and trajectory information available at nodes.
- **Per-hop behavior:** Unlike existing protocols, AeroRP forwards data per-hop based on partial local information and routes thereby avoiding the necessity for global convergence, making it especially suitable for highly-dynamic environments.

- **Multi-modal:** Military applications present a high level of variation in their operational parameters. For example, based on the security requirements of the test application, the geolocation of the nodes may or may not be available. In order to support these dynamics in operation, policies, and constraints, AeroRP provides multiple modes of operation.

Table 4.4: Feature comparison of AeroRP and other routing protocol categories

Feature	**AeroRP**	**MANET** (AODV, OLSR, DSDV, DSR)	**Opportunistic** (OR, EOR)	**Geographic** (LAR, DREAM)	**Beaconless** (IGF, BOSS)
partial-paths	yes	no	yes	yes	yes
store & haul	yes	no	no	no	no
cross-layering	yes	no	no	yes	yes
snooping	yes	no	no	no	yes
location aware	yes	no	no	yes	yes
beaconless	optional	no	no	yes	yes
update frequency	aperiodic topology dependent	periodic or on-demand	no updates	periodic	no updates
route reconfiguration	hop-by-hop	source initiated or based on updates	hop-by-hop	based on updates	hop-by-hop
multi-mode	yes	no	no	no	no

Protocol Operation

The basic operation of AeroRP consists of two phases. In the first phase, each AN learns and makes a list of available neighbors at any given point in time. It utilizes a number of different mechanisms to facilitate neighbor discovery, discussed later in this section. The second phase of the algorithm is to find the appropriate next hop to forward the data packets. In order to forward the packets toward a specific destination, additional information such as location data or route updates is required. For each of these two phases the protocol defines a number of different mechanisms. The particular choice of mechanism to be used is dependent upon the mode of operation. The protocol does not specify a predefined set of discrete operational modes; the total number of supported

modes is merely the combination of all the different mechanisms available. We now consider each of the two phases in more detail:

Neighbor Discovery: The first objective of an airborne node is to determine its neighboring nodes. In order to achieve this, we use several different mechanisms with the objective to minimize overhead and increase adaptability. One or more of the following mechanisms may be used to populate the forwarding table depending upon the operational constraints.

- ***Active snooping*** is the primary mechanism used by the node to locate and identify its neighbors. In the wireless TDMA network, a node that is not transmitting listens to all transmissions on the wireless channel. AeroRP adds the transmitting MAC address of each overheard packet to its neighbor table. The protocol assumes cooperative nodes and symmetric transmission ranges among ANs. This implies that if a node can hear transmissions from a node, it can also communicate with that node. Stale entries are removed from the neighbor table if no transmissions from a node are heard for a predetermined time interval related to the anticipated contact duration.

- ***Hello beacons*** are used by idle nodes to advertise their presence. When neighboring nodes hear a hello beacon, they update their neighbor table appropriately. The frequency of the hello beacon is inversely proportional to the minimum calculated contact duration. For example, if the minimum contact duration is 10 s, the hello beacon is transmitted every second however if the minimum contact duration is 100 s, the hello beacon need only be sent every 10 s.

- ***Ground station updates*** may used to augment or replace *active snooping* in some of the mission scenarios, in which the ground station has a partial or even

complete mission plan. The ground station sends periodic updates containing the location and trajectory vectors predicted by the mission plan to all nodes.

Security requirements may impose certain restrictions on aeronautical networks. In certain cases in which node location or trajectory is considered sensitive, individual nodes may not include this information in the header of data packets or hello updates. In this case, the ground station may send location updates of all nodes on an encrypted channel. Finally, in the most secure mode, no geographic node information is available and the routes have to be discovered using traditional MANET methods, such as explicit routing updates and the exchange of node contacts between neighbors.

Given the dynamic nature of the aeronautical network, neighbor discovery not only consists of finding nodes within transmission range, but also determining the duration for which a discovered node will remain within range. Depending upon operational constraints, this information is obtained via different mechanisms: location and trajectory information is included in the AeroNP header [243], or in updates sent by the ground station.

Data Forwarding: After neighbor discovery, the second phase of AeroRP is for individual nodes to determine the next hop for a particular transmission. Recall that, unlike conventional protocols, AeroRP performs hop-by-hop forwarding based on partial paths without the full knowledge of the end-to-end paths [279]. Each node forwards packets such that they end up geographically closer to the destination, which will frequently be a GS in many mission scenarios.

When any given node needs to transmit data, and assuming that one or more neighbors are discovered, the data packets are forwarded to the node that is nearest to the destination as calculated from its current coordinates and trajectory. The destination location is obtained in a manner similar to that of discovering neighbors. Furthermore, in many

cases the destination is the stationary ground station whose coordinates are known to all ANs. The algorithm for finding the best node to forward (or handover) the data packet is given in Section 3.2.4

In order to avoid congestion at any given node, AeroRP utilizes the *congestion indicator* (CI) [280, 281] field of the AeroNP header. Each node uses the *CI* field to indicate its own congestion level. All packet transmissions from a node carry the *CI* field along with the type and priority of the data. All the neighboring nodes are thus made aware of the congestion at a given node for a given priority of the traffic and refrain from forwarding equal or lower priority traffic to the congested node.

Ground stations are special nodes in this network. They listen to all the transmissions and forward packets that are destined to other GSs. In other words, GSs are universal sinks and may share the same MAC address. For uplink data, a GS forwards data to the node that is closest to the destination node. The GS is aware of the location of all nodes either from mission planning or by learning it during the test from header information in received packets.

RNs (relay nodes), if present, are always the default next-hop. They accept packets from all the ANs and forward them directly to the ground station or another AN. Since the GS has narrow beam width and can only track one AN at a time, it is more efficient for the GS to track RNs and have individual ANs forward their data via RNs. Given the varied service requirements of tactical missions, AeroRP supports multiple modes for both open and secure scenarios.

Mission Based Quality of Service

The wireless links in the telemetry network are bandwidth-constrained and may be under-provisioned for the traffic generated at any give time. Hence, it is essential to implement

a quality of service mechanism in this network to ensure that high priority data, such as command and control, can be reliably delivered. The AeroNP protocol uses two fields in the header to specify the quality of service of packets in the network: data *type* (e.g. command and control, telemetry) and *priority* within a given type. The mission and application requirements determine the *type* and *priority* for a given data flow, which are passed to AeroNP through AeroTP via out-of-band signaling. The scheduling algorithm at nodes is weighted fair queuing based on type and priority.

Broadcast and Multicast

The AeroNP protocol supports both broadcast and multicast natively. The typical all-ones MAC address is used as the broadcast address. Similarly, a range of MAC addresses are assigned to sub-groups in the network. These multicast address groups are generally pre-programmed in the nodes and GS. Note however, that given the highly dynamic nature of the network multicast may not achieve any significant benefit over a simple broadcast in terms of efficiency for sparse networks.

Congestion Control

In a heavily loaded network multihop routing can induce severe congestion at nodes involved in multihop forwarding as well as transmitting their own telemetry data. To overcome this, AeroNP uses a simple congestion control mechanism at the network layer using *congestion indicators* and *back pressure*. We choose these algorithms for their simplicity in their operation based on little feedback. The objective is to avoid local congestion and it does not guarantee global optimization or fairness. A more rigorous rate control mechanism such as one proposed in [282] is not suitable here due to the highly dynamic nature of the network, in which an optimal solution would become stale by the time it is achieved.

In the first mechanism, the node uses the *CI* (congestion indicator) [280, 281] field to indicate its own congestion level. Even though 2 bits are assigned to *CI* field, only two of the four possible values are currently used. Hence *CI* is toggled between `0x0` and `0x3`. All packet transmissions from a node carry the *CI* field along with the type and priority of the data. When the transmit queue of a node exceeds a predetermined threshold, the node sets its *CI* field to `0x3`. Neighboring nodes eavesdrop on the transmission and are made aware of the congestion at a given node. If a node is congested, the neighbors back off if the data that they have is of equal or lesser priority, however higher priority data is still forwarded to a congested node because the priority queue at that node will service this traffic first.

The second mechanism through which congestion control is achieved in the telemetry network is *back pressure* [132, 283]. As a source sends packets to an intermediate node, it simultaneously eavesdrops on that node to see if the packets are being forwarded at the same rate they are being sent. If not, and other packets are being forwarded instead, then the source can infer that the next hop it has chosen is queuing its packets due to congestion. The source node then backs off and if possible chooses an alternate next-hop. Similarly, in a multihop scenario, if a bottleneck is encountered, each intermediate hop either stops or slows down its transmissions on the congested path successively until the source of the traffic is reached.

4.3 AeroTP Simulation Model

In this section we describe the ns-3 simulation model of AeroTP. The purpose of this model is to allow us to compare its performance with other transport protocols, as well as refine its performance.

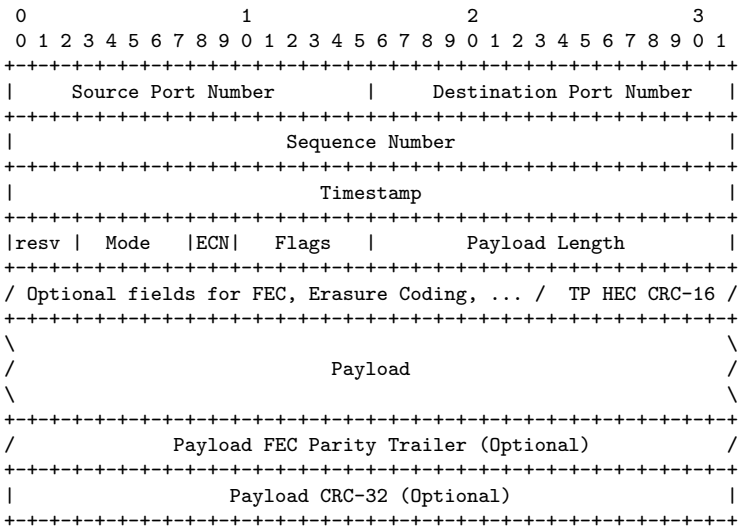

Figure 4.20: AeroTP data segment structure

4.3.1 AeroTP Segment Structure

An AeroTP segment (shown in Figure 4.20) is structured for a bandwidth-constrained network so it does not encapsulate the entire TCP/UDP and IP headers, but is capable of converting to the TCP/UDP format at the gateways. To satisfy the end-to-end semantics it keeps certain fields in common with the TCP/UDP headers such as the source-destination port numbers, TCP flags, and the timestamp. The sequence number uniquely identifies AeroTP segments for reordering them at the receiving edge and for error-control purposes. The HEC (header error check) field performs a strong CRC on the header to detect bit errors caused by wireless channel, thus making sure the packet is correctly transmitted to the destination. In case the payload gets corrupted, AeroTP performs FEC on the payload. The payload CRC is used in the absence of a link-layer frame CRC and enables the measurement of bit-error-rate for error correction depending on the transfer mode used.

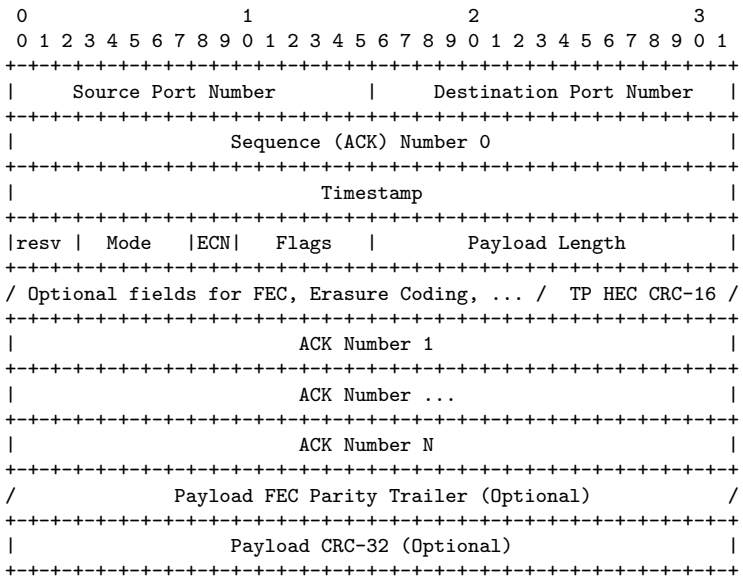

Figure 4.21: AeroTP MACK segment structure

4.3.2 AeroTP Operation

As a connection-oriented protocol, it is essential to define and maintain consistent states at the sender and the receiver to establish a connection for data transfer. The states either remain the same or evolve to another depending on the events and actions that happen within the protocol during a communication session. Figure 4.22 shows the AeroTP reliable mode packet flow-diagram, in which S is the source, D is the destination, and TmNS represents the telemetry network and Figure 4.23 shows the state transition diagram.

Similar to TCP, the AeroTP source-destination pair uses control messages (ASYN, ASYNACK, AFIN, and AFINACK) for opening and closing a connection. The difference is an opportunistic connection establishment, in which data and control overlap, is chosen over the TCP's three-way handshake, thus saving a round-trip-time that is otherwise wasted. Initially the sender and the receiver are in the CLOSED state. A connection is initiated by the sender through an APP_CONNECT message from the application. The sender then

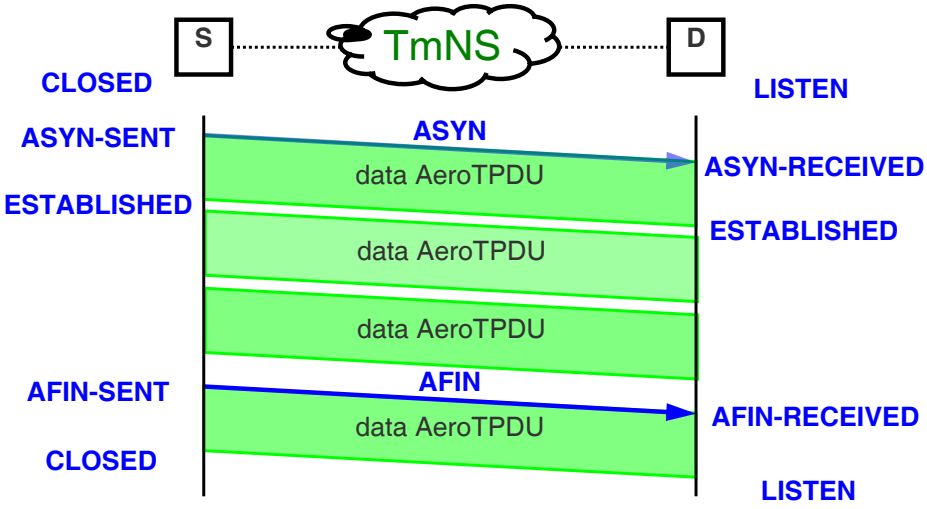

Figure 4.22: AeroTP connection management

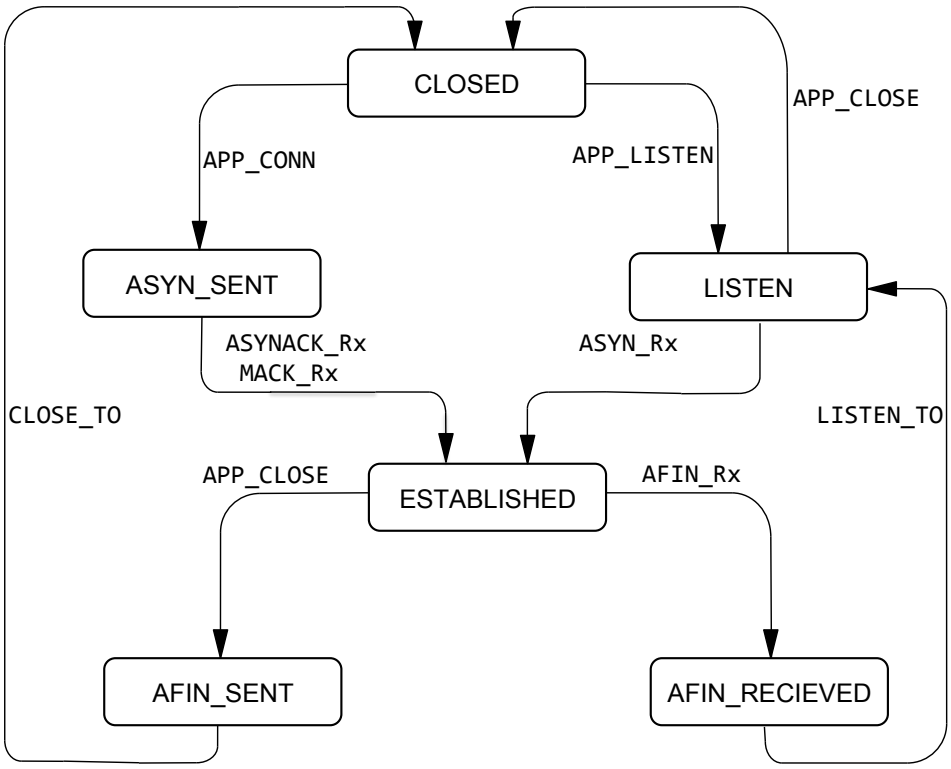

Figure 4.23: AeroTP state transition diagram

Table 4.5: State & Transition Definitions

State	Description
CLOSED	No connection & no data is transferred
LISTEN	Receiver is ready to listen to any incoming data
ASYN_SENT	ASYN message sent by the host initiating connection
ESTABLISHED	A steady state in which data transfer takes places
AFIN_SENT	AFIN message sent to indicate no new data being sent
AFIN_RECEIVED	AFIN message received
APP_CONN	Connection request issued by application
APP_LISTEN	Listen request issued by application
APP_CLOSE	Close request issued by application
ASYN_RX	ASYN received
ASYNACK_RX	ASYNACK received, connection est.
MACK_RX	Single or multiple packet ACK received
AFIN_RX	AFIN received, indicating end of any new data
CLOSE_TO	A timeout before going to the CLOSED state
LISTEN_TO	A timeout before going to the LISTEN state

sets the ASYN bit in the AeroTP header and transmits it with or without data depending on the data being available in the send buffer and moves to the AFIN_SENT state. The receiver receives an APP_LISTEN request from the application and moves to the LISTEN state, and upon receiving the ASYN message it moves to the ESTABLISHED state, and acknowledges the ASYN by setting the ASYN bit and the MACK bit simultaneously. The sender moves to the ESTABLISHED state as long as the ASYNACK or a simple MACK is received, so that the connection does not have to terminated in case the ASYNACK gets lost. While in the ESTABLISHED state, the sender and the receiver exchange AeroTPDU and ACKs. After the sender is finished with sending all the data, an AFIN message (with AFIN bit set in the header) is sent to the receiver and the sender moves to the AFIN_SENT state. Upon receiving the AFIN message the receiver moves to the AFIN_RCVD state and transmits an AFINACK. To make sure that the receiver has acknowledged all the data including retransmissions, the receiver begins a timer in AFIN_RCVD state which goes to a LISTEN state after a time long enough so that all the retransmissions have likely been received from the sender. The sender also maintains a close timer, which expires after a certain time interval that is long enough to likely receive acknowledgements for all data packets. This way the sender is guarantees delivery of all the data packets with high probability.

In the reliable mode, error-control is achieved by implementing a selective-repeat ARQ mechanism. To reduce the overhead incurred, AeroTP aggregates multiple ACKs at the receiver into a single packet before transmitting them to the sender, as TCP does using the SACK option [89]. The number of ACKs to be aggregated is a tunable parameter, and the optimal value depends on the probability of error or loss in the channel, as well as the rate at which packets are sent. AeroTP also uses a timer to guarantee that ACKs will not be delayed long enough to trigger unnecessary retransmissions.

4.4 Simulation Results

This section presents results from simulations of the AeroTP and AeroRP protocols performed using ns-3 and ns-2 respectively. The performance of AeroTP is compared to TCP, and the performance of AeroRP is compared to that of DSDV and AODV.

We have implemented ns-3 models of the fully-reliable (ARQ) and quasi-reliable (FEC) modes described in Section 4.3. This section presents the simulation results from running these models. We compare the performance of AeroTP in the reliable connection and quasi-reliable modes with TCP and UDP protocols using the ns-3 open-source simulator [284]. The selective-repeat ARQ algorithm is used to provide reliable edge-to-edge connection between nodes for the reliable mode, and FEC (as discussed above) is used for the quasi-reliable mode of the AeroTP protocol. The network in this simulation setup consists of two nodes communicating via a link that is prone to losses. One node is configured as a traffic generator, and the other as a traffic sink. The traffic generator sends data at a constant data rate of 4.416 Mb/s (3000 packets/s with an MTU of 1500 B). The path consists of a 10 Gb/s link representing the LAN on the TA, a 5 Mb/s link with a latency of 10 s representing the mobile airborne network, and a second 10 Gb/s link representing the LAN at the ground station. Bit errors are introduced in the middle

link with a fixed probability for each run, and the performance for each probability of bit-errors is shown in the plots described in the next section. A total of 1 MB of data is transmitted during one single simulation between the two nodes. The link is made unreliable by introducing losses using an error model varying bit-error probabilities ranging from 0 to 0.0001 for each of the protocols. Each simulation case is run 20 times and the results averaged to obtain the data needed for comparison.

4.4.1 AeroTP Connection Establishment

As mentioned previously, one of the drawbacks of TCP for highly-dynamic airborne environments is the three-way-handshake used for connection establishment. For this reason AeroTP is designed to establish a connection when the first data TPDU (with ASYN bit set) in a flow is received. If the first packet is lost, the connection can still be established using header information from the second or subsequent data packet, and the first packet can be retransmitted later if required by the specified reliability mode. To illustrate the difference between these two approaches, we have done simulations comparing the time required to establish a standard TCP connection, compared to a AeroTP connection.

The simulations are implemented in the ns-3 open-source simulator [284]. Each simulation consists of two nodes connected by a 10 Mb/s link with 5 ms latency and a fixed probability of packet loss, which is varied between 0 and 20% as seen on the x-axis. Node 0 is configured as a traffic generator (TCP or AeroTP as appropriate) and node 1 is configured as a traffic sink. For each packet-loss probability point plotted, the simulations were run 100 times and the results averaged. Each simulation consists of a single connection attempt by either TCP or AeroTP. We record the delay starting when the `connection_setup` command is issued to the transport protocol, and stopping when the first *data* packet is received by the data sink.

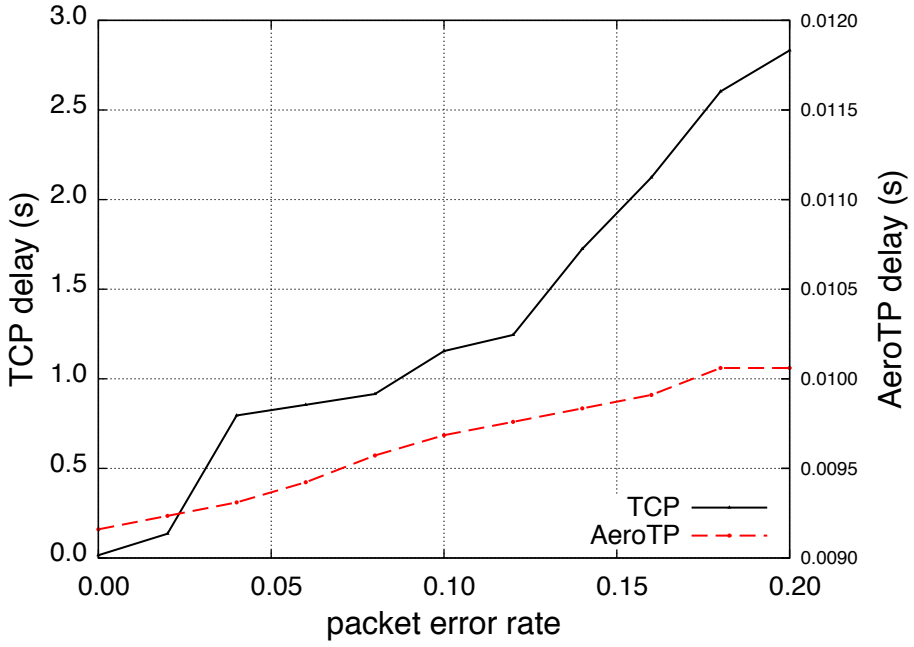

Figure 4.24: TCP and AeroTP connection establishment delay

Figure 4.24 shows the results of these simulations. Both the TCP and AeroTP results are presented in a single plot, however, note that they are plotted against two different y-axes: TCP on the left, and AeroTP on the right. The TCP delay starts at about 20 ms when no losses occur, and increases linearly until it approaches 3 s when the packet-loss rate is 20%. AeroTP on the other hand, has a delay of 9.2 ms when no losses occur, and increases linearly to 10.1 ms when the packet-loss rate is 20%. This shows an improvement of two orders-of-magnitude, which will play a large role in enabling AeroTP to successfully send data over paths which only exist for a few seconds, while TCP would still be trying to establish the connection.

4.4.2 Fully-reliable mode performance

In *fully-reliable* mode, AeroTP uses ARQ as its reliability mechanism. This trades off additional latency (in the case of lost or corrupted packets) and overhead of the reverse

channel, for reliability. The advantage to this mechanism is that given enough time, every lost packet can be retransmitted. In our model we are able to adjust the amount of bandwidth required by adjusting the number of ACKs aggregated into a single packet before it is transmitted. We found this to have a negligible effect on performance, and so have omitted the set of plots showing adjustments to this parameter to save space. The overall performance of the fully-reliable mode can be seen in our last set of plots.

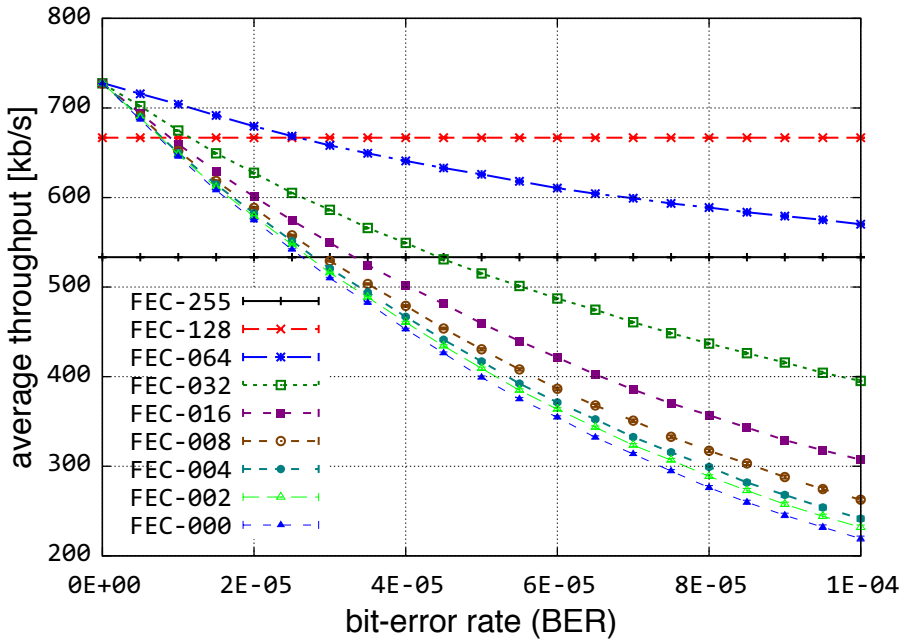

Figure 4.25: Average goodput

4.4.3 Quasi-Reliable Mode Characterization

In *quasi-reliable* mode, AeroTP uses FEC as its reliability mechanism. This trades overhead on each packet for reliability. The advantage to this mechanism is low delay, because no retransmissions are required to correct errors. Our first set of plots compares varying FEC strengths, from zero FEC 32 bit words per packet, to 256 FEC words. In all cases 1500-byte packets are used, thus as the number of FEC words in each packet is increased, the number of bytes of data in each packet decreases, meaning that more

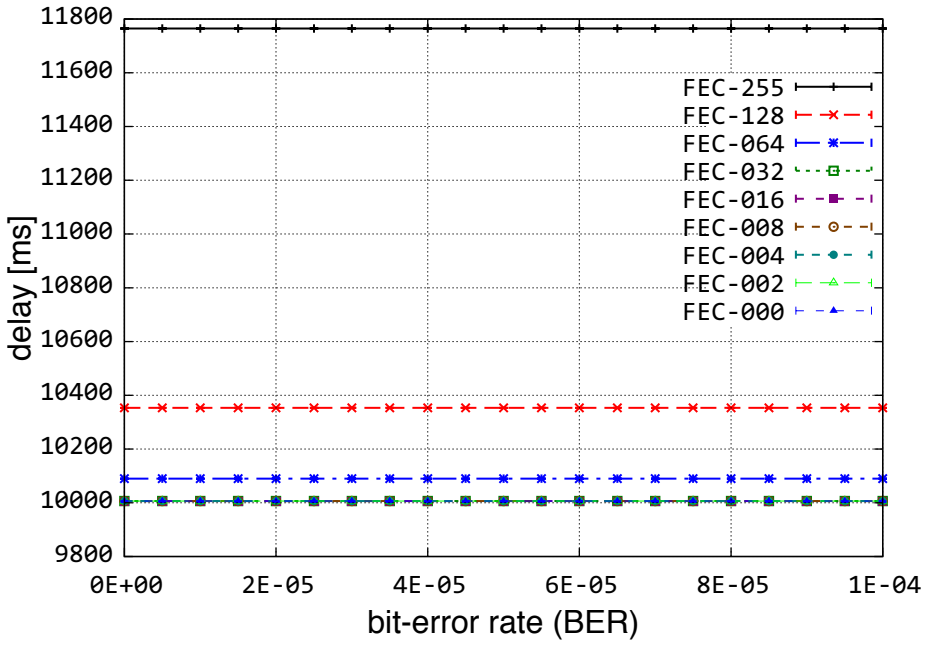

Figure 4.26: Average delay

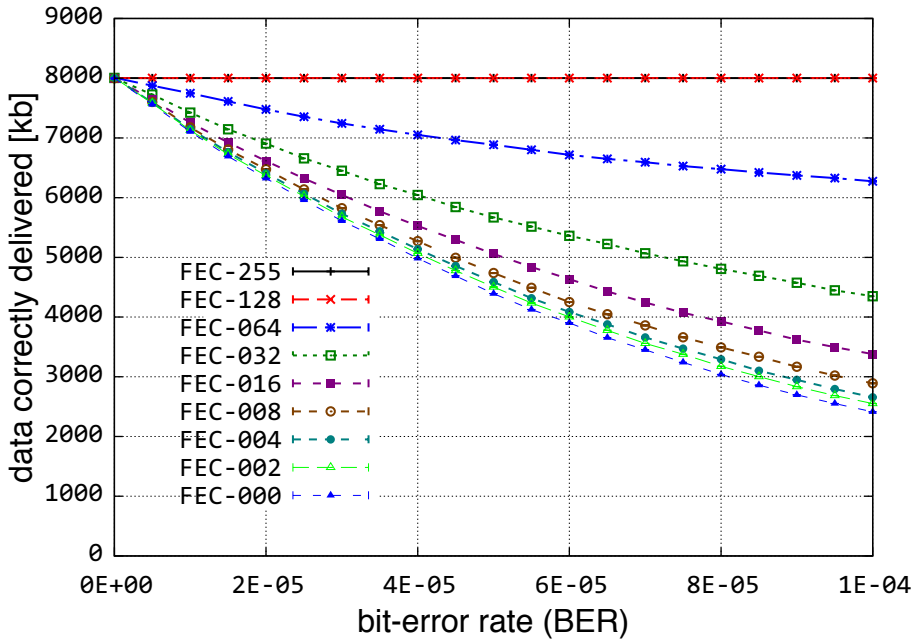

Figure 4.27: Cumulative goodput

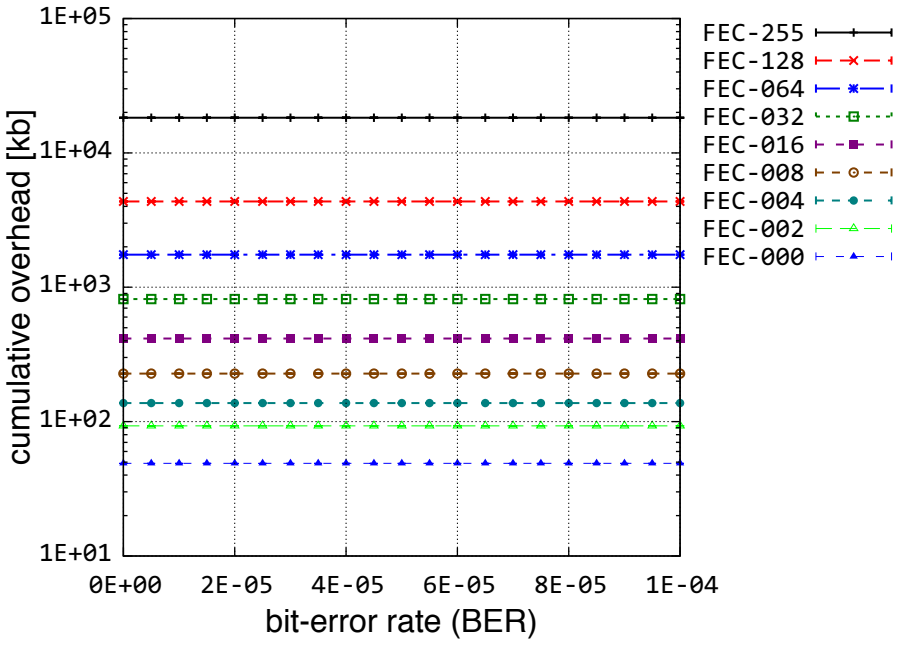

Figure 4.28: Cumulative overhead

packets are required to transfer the same amount of data. Figure 4.25 shows that the throughput decreases due to uncorrectable packets as the error-rate increases, however this effect can be mitigated by increasing the FEC strength. For very high FEC strengths (128 and 256), there was virtually no decrease in performance across the range of error-rates tested, however the performance is decreased at low error-rates due to the high level of overhead. Due to the fact that retransmissions are not involved, the latency of data transmission is not affected by packet errors as shown in Figure 4.26. However, as very high levels of FEC result in link saturation this translates into added latency due to queuing delay. Similar to the throughput plot, Figure 4.27 shows the total amount of data transmitted. Depending on the FEC strength, this quantity decreases as the error-rate increases, except for very high FEC strengths (128 and 256) all errors are able to be corrected, at the rates tested. Lastly in this set of plots, we show the overhead imposed on the network by using the FEC mechanism at various strengths (Figure 4.28). This quantity is significant (note the log y-axis scale), however it is not affected by the

error rate.

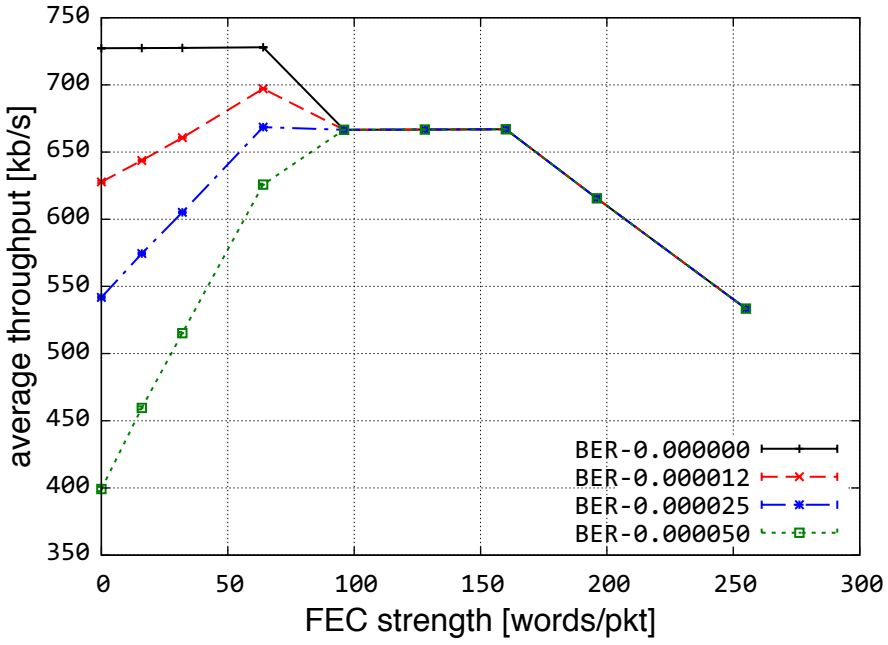

Figure 4.29: Average goodput

The next set of plots continue to characterize the *quasi-reliable* mode. Figure 4.29 shows that for error rates greater that zero, higher FEC strengths result in higher throughput up to a point. For the 128-word and 256-word FEC strengths, the amount of FEC bytes being sent begins to saturate the link, resulting in reduced throughput of data. Figure 4.30 shows that for all error rates, higher FEC strengths increase delay slightly as the link becomes saturated. Figure 4.31 shows that an FEC strength of 96 words/packet or greater is able to correct all errors at the error rates tested. Figure 4.32 shows the increase in overhead resulting from increased FEC strength. The increase is linear with respect to the amount of data being sent, but since we have chose to quantify FEC strength with respect to the number of packets transmitted it appears exponential, due to the fact that an increase in FEC strength results in an increase in the number of packets sent to transmit a give amount of data using maximum-size packets. Future

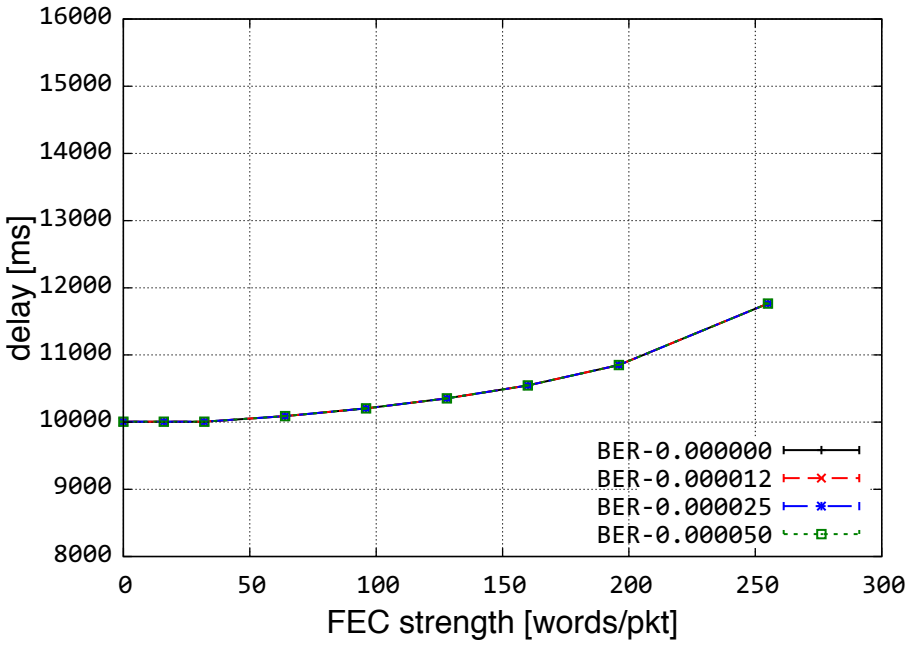

Figure 4.30: Average delay

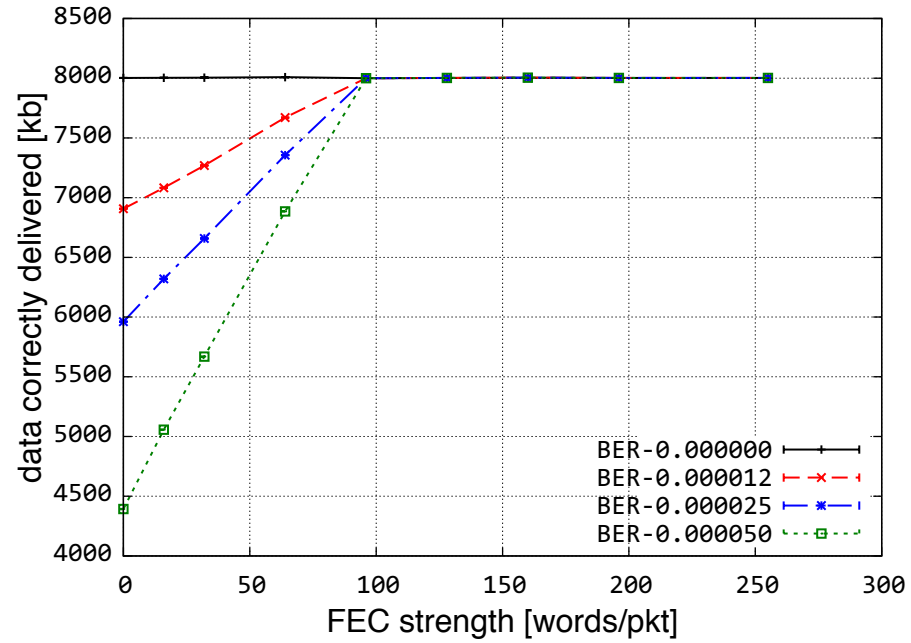

Figure 4.31: Cumulative goodput

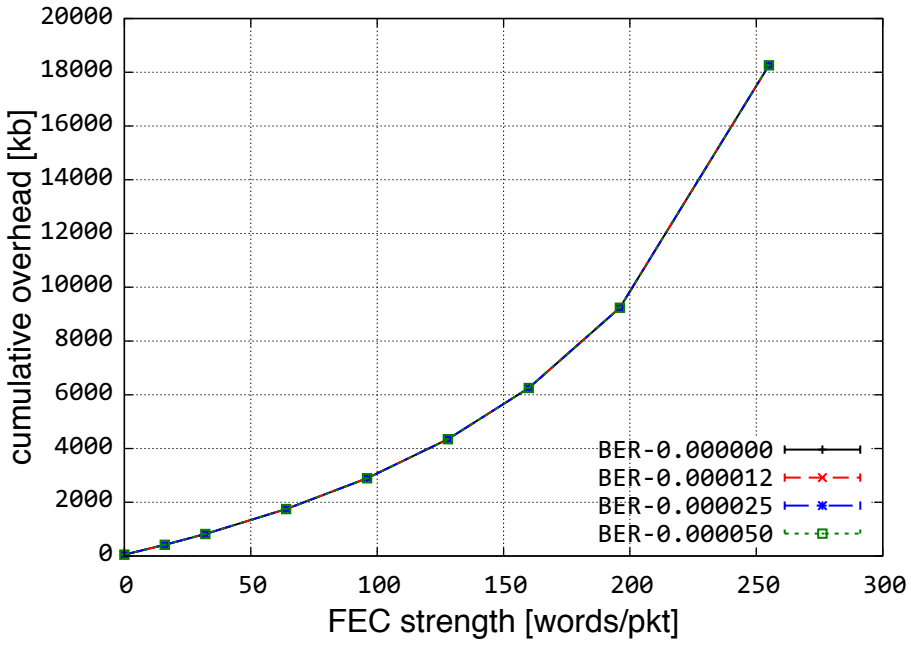

Figure 4.32: Cumulative overhead

models may change this quantification in favor of one relative to the amount of data being transmitted.

4.4.4 Performance Comparison over Lossy Links

Figure 4.33 shows that AeroTP reliable-mode is able to achieve significantly better performance than TCP, which backs off substantially as the BER (bit-error rate) increases. TCP also becomes highly unpredictable in its performance, as shown by the error bars. At the same time TCP's end-to-end delay increases by 3 orders-of-magnitude doubles with a BER of 1×10^{-4}, while AeroTP increases less than 1 order-of-magnitude as shown in Figure 4.34. Over the course of the simulation, both TCP and AeroTP are able to deliver the full 1 MB of data transmitted for low error rates <0.000035, but above that TCP performance drops rapidly while AeroTP is still able to deliver nearly all the data at the highest error rates as shown in Figure 4.35. In the same plot we see that UDP looses

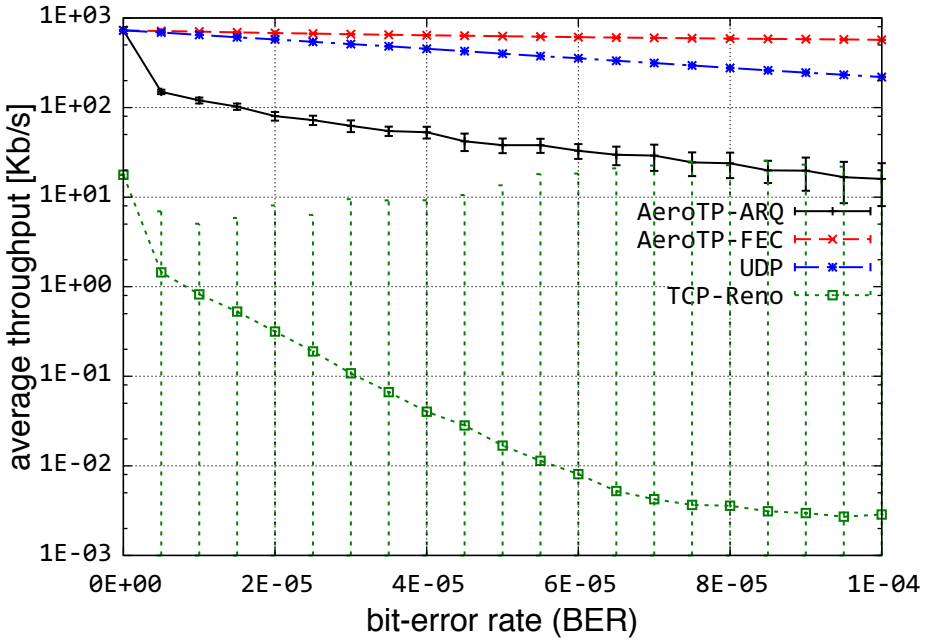

Figure 4.33: Average goodput

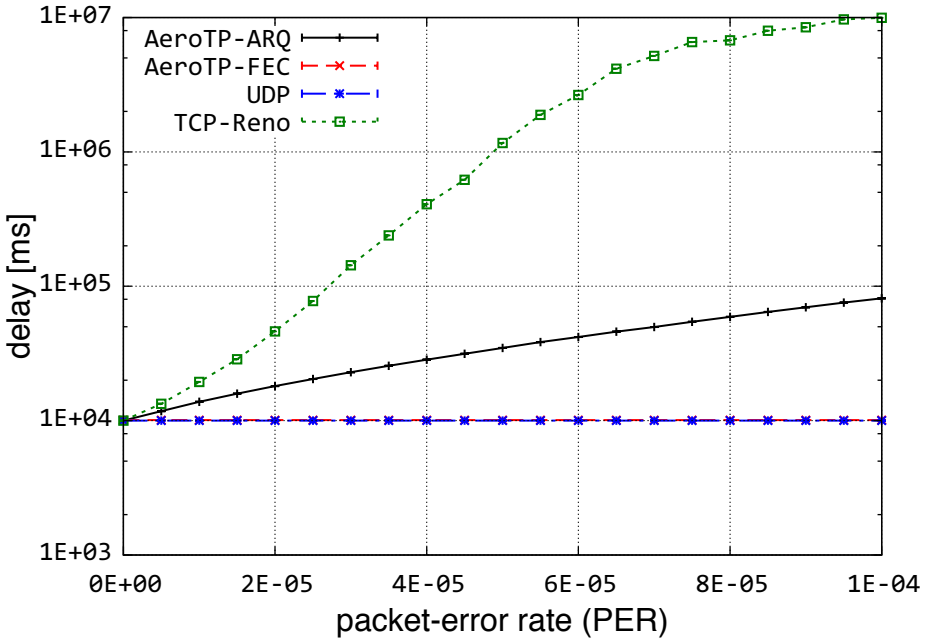

Figure 4.34: Average delay

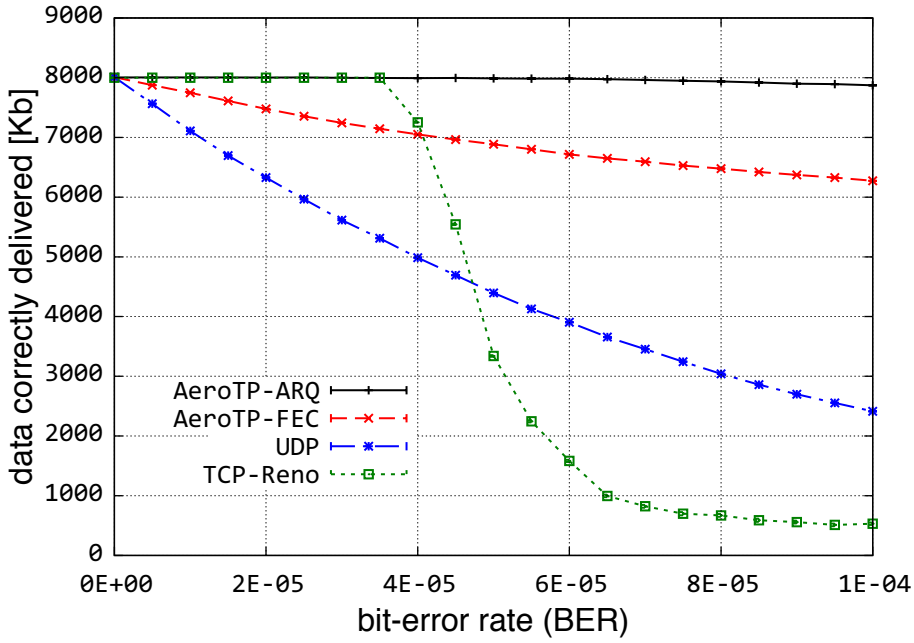

Figure 4.35: Cumulative goodput

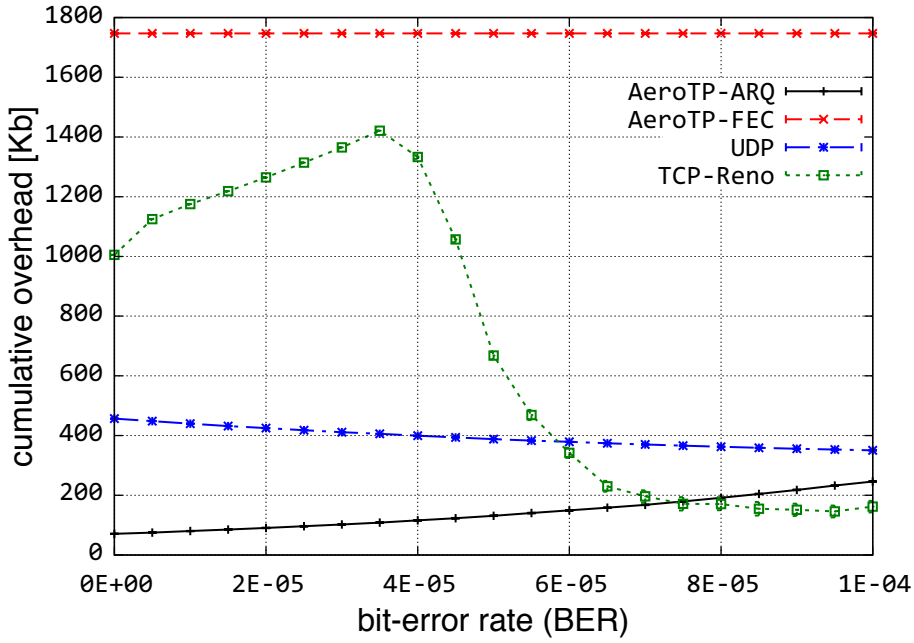

Figure 4.36: Cumulative overhead

a percentage of the data due to corruption as the BER increases, and that the AeroTP quasi-reliable mode losses a much smaller percentage. Lastly in Figure 4.36 we see that the performance improvement of the AeroTP reliable-mode is achieved with much lower overhead than TCP, while quasi-reliable mode does incur significant overhead, but does not cause any increased delay as the BER increases.

4.5 Summary

In this chapter we presented the design of two transport protocols, ResTP and AeroTP, and evaluated the improvement in performability enabled by using path diversification, opportunistic connection setup, decoupling loss from flow-control, and the tradeoff between closed and open-loop error control. AeroTP is domain specific, and designed to work in conjunction with the rest of the ANTP suite that we designed for highly-dynamic airborne environments. For context we have provided an overview of the routing and network layer components of this suite, and shown how AeroTP also provides increased performability in this environment through increased packet delivery ratio.

In the next chapter we will present the design of the GpENI (Great Plains Environment of Network Innovation) network testbed that we have built in collaboration with many institutions worldwide, in order to support at-scale protocol implementations at any network layer. GpENI is the platform being used to implement and validate the ResTP protocol as well as the entire ANTP suite (AeroTP, AeroNP, and AeroRP).

Chapter 5

GpENI Testbed

In order to test multilayer resilience protocols at scale, we have developed the GpENI testbed, which is now being utilized to evaluate prototype versions of AeroTP and the ANTP protocols, with ResTP soon to follow. This chapter describes the planned experiments, the testbed, and the challenges faced in its development. The work described in this chapter has resulted in several publications. The ANTP prototype architecture being implemented on GpENI is described in [285], and the design of the GpENI testbed is layed out in [286]. A current status of the development of GpENI and the challenges encountered is discussed in [287] and future experiments on path diversity are described in [288].

5.1 Current ANTP Prototyping

We are using Python as the prototyping language for the ANTP protocols, both to make efficient use of available programmer cycles, as well as for its cross-platform support. While the initial testing is being performed on the GpENI testbed, we will also be testing on radio-controlled vehicles with embedded linux platforms. The current work, which is being performed by several members of the ResiliNets group, includes several utility

programs including a GPS emulator for GpENI nodes, so that it can provide position data to AeroRP, and coupled with that is a wireless broadcast emulator that determines which nodes overhear a transmission based on position and a random function. We have also implemented a central data collection system to monitor the progress of each experiment, with a web-based map overlay showing the emulated position of each node and giving access to AeroRP interface counters and forwarding tables. Along with data collection we also have a server that will provide experiment control instruction to each of the nodes, although the full specification of these controls is not yet completed. At the transport layer, we have been successful in implementing both the ARQ (fully-reliable) and FEC (quasi-reliable) modes. FEC uses existing Reed-Solomon error-correction libraries available in Python, and likewise both modes use existing CRC error checking Python functions.

Each component of the ANTP suite uses separate threads for communicating with the server, so that the experiment control and monitoring will have minimal impact on the protocol performance. Due to GpENI security restrictions (in the PlanetLab component described later) all ANTP packets are encapsulated inside UDP packets, which while not affecting the function does add some overhead.

5.2 Planned ResTP Experiments

Once the KU challenge emulation infrastructure is in place on GpENI, numerous ResTP flows will be established, based on the Path Diversification metrics based using the geographic coordinates of intermediate nodes. In addition to using multiple diverse paths, we will be evaluating the performance of several end-to-end and hop-by-hop reliability mechanisms, such as FEC, erasure coding, and ARQ. With the traffic model active, we will then perturb the experiment to cross-validate our simulation findings by evaluating

the level of resilience of each topology, as well as the resilience of each reliability mechanism, in the face of challenges. The challenges will parallel those introduced by the ns-3 challenge model, including random failures, targeted attacks, and large-scale disaster affecting a particular geographic region. The data collected from these experiments will allow us to relate theoretical graph properties to real-world performance benefits, as well as to quantify the improvements afforded by particular end-to-end reliability mechanisms and combinations of mechanisms.

5.3 GpENI Overview

The Great Plains Environment for Network Innovation – GpENI is an international programmable network testbed centered on a regional optical network between The University of Kansas (KU) in Lawrence, Kansas State University (KSU) in Manhattan, University of Nebraska – Lincoln (UNL), and University of Missouri – Kansas City (UMKC) within the Great Plains Network, supported with optical switches from Ciena interconnected by Qwest fiber infrastructure, in collaboration with the Kansas Research and Education Network (KanREN) and Missouri Research and Education Network (MOREnet). GpENI is undergoing significant expansion to Europe and Asia. The goals of GpENI are to:

- Build a collaborative research infrastructure in the Great Plains region among GPN and other institutions
- Construct an international programmable network infrastructure enabling GpENI member institutions to conduct experiments in Future Internet architecture, supporting projects such as PoMo: PostModern Internetwork Architecture [24] and ResumeNet [289], including evaluation of ResTP and path diversity mechanisms

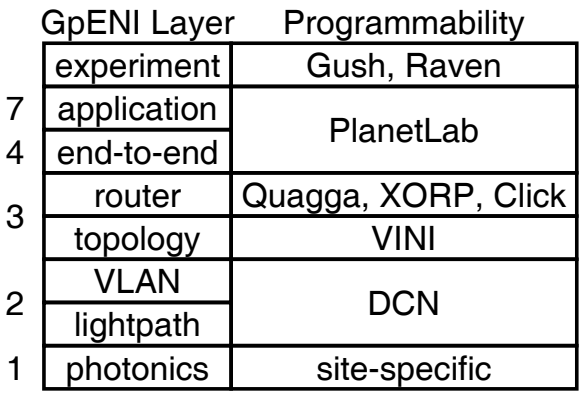

Figure 5.1: GpENI Programmability Layers

- Provide programmable optical infrastructure to the GpENI Midwest optical backbone, and expand optical connectivity to selected international sites
- Provide flexible infrastructure to support the GENI program as part of the PlanetLab control framework cluster B
- Deploy tools developed by GpENI and the GENI community such as Gush for experiment control and Raven for code deployment
- Provide an open environment for networking research community experiments

5.3.1 GpENI Programmability and Flexibility

The defining characteristic of GpENI is programmability of *all* layers, as shown in Figure 5.1, implemented on a *node cluster* of general- and special-purpose processors, described in detail in Section 5.5. At the top layer Gush provides experiment control and Raven distributes code; both are software developed as part of the GENI program. Layer 7 and 4 programmability are provided by the GENIwrapper version of PlanetLab. At layer 3, programmable routers are implemented in Quagga, XORP, and Click, supplemented by any other technology for which GpENI institutions should choose to

deploy.[1] Flexible network-layer topologies are provided by VINI. At layer 2, dynamic VLAN configurations are provided by DCN-enabled managed Gigabit-Ethernet switches at the center of each GpENI node cluster. GpENI institutions directly connected to the optical backbone use DCN-enabled Ciena switches to provide dynamic lightpath and wavelength configuration. At layer 1, the architecture even permits programmability at the photonic layer for switches that provide such support. Furthermore, each GpENI institution can connect site specific networking testbeds; plans include wireless, sensor, and cognitive radio testbeds (e.g. KUAR [291]). External users in the broader research community may request GpENI accounts with which to run network experiments.

5.3.2 GpENI Suitability for ResTP

Due to the role that geography and topology play in this work, making the transition from simulation to testbed poses unique infrastructure challenges, especially since it is desirable to run prototypes under many different traffic and large-scale topology scenarios. It is also desirable to have a heterogeneous testbed, with a geographically distributed backbone topology to emulate tier-1 service providers, as well as more localized clusters to emulate regional access networks and subscriber nodes along with user traffic load. Since we are concerned with full end-to-end dependability it will be necessary to include all of these components within the testbed scenario.

To meet these requirements we have identified a number of components within GENI that are suitable for different aspects of our experimentation. The OpenFlow infrastructure installed on Internet 2 (and planned for selected GpENI node clusters) provides a configurable large scale backbone in North America. The GpENI [1] infrastructure using the Dragon Dynamic Circuit Network (DCN) [292] flexible VLAN topology extends geographic coverage to Europe and Asia, as well as connecting the G-LAB [293] infras-

[1] We are investigating including OpenFlow [290] as a core GpENI technology.

tructure in Germany. These three infrastructures will allow a sufficiently large number of geographically-distributed topology overlays to be formed, based on output from the KU-LocGen topology generator. The second tier of experimental infrastructure will be used to model access networks and end-users. Most GpENI clusters (which implement the PlanetLab control framework [294]) have only a few nodes, so federating with EMULAB [295] (using the ProtoGENI control framework [296], via the GpENI–ProtoGENI physical connection in the Kansas City I2 PoP) and G-LAB clusters will be necessary to enable full end-to-end flow prototyping over a realistic hierarchical infrastructure.

5.4 Topology and Network Infrastructure

The core of GpENI is the regional optical backbone centered around Kansas City. This is extended by KanREN (Kansas Research and Education Network) to various GPN (Great Plains Network) institutions in the Midwest US. Connectivity in Kansas City to Internet2 provides tunneling access to the Canadian, European, and Asian GpENI infrastructure. Optical connectivity is currently in place in the UK between Lancaster and Cambridge, and will replace other GpENI L2TPv3 and IP tunnels as available. GpENI is growing, currently with about 200 nodes at 40 institutions in 20 nations. Institutions may connect to GpENI if they are interested in becoming part of the GpENI community, and have the resources to install, connect, and manage a node cluster.

5.4.1 Midwest US GpENI Core and Optical Backbone

GpENI is built around the core GpENI optical backbone centered in the Midwest, shown in Figure 5.2, among the principal institutions of KU, KSU, UMKC, and UNL, currently under extension to additional institutions, including the GMOC (GENI Meta-Operations Center). The optical backbone consist of a fiber optic run from KSU to KU to the

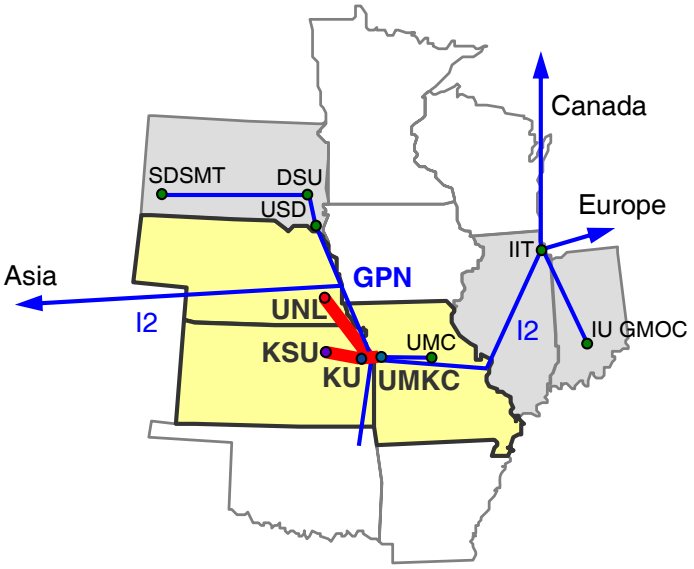

Figure 5.2: GpENI Midwest Topology

Internet2 PoP in Kansas City, interconnected with dedicated wavelengths to UMKC and UNL.

Each of the four core institutions will have a node cluster that includes optical switching capabilities provided by a Ciena CoreDirector or CN4200, permitting flexible spectrum, wavelength, and lightpath configurations.[2]

5.4.2 GpENI International Topology

GpENI is extended to Europe across Internet2 to GÉANT2 and NORDUnet and then to regional or national networks, as shown in Figure 5.3. Currently, connectivity is achieved using L2TPv3 and IP tunnels. A direct fiber link over JANET is deployed between Lancaster and Cambridge Universities.[3] The principal European GpENI institutions are Lancaster University in the UK and ETH Zürich in Switzerland. Similarly, GpENI is

[2]GpENI optical infrastructure is currently undergoing phased deployment; the UNL switch is installed and the KU switch is under procurement.

[3]GpENI is currently working with research network providers to increase direct optical connectivity.

extended to Asia across Internet2 to APAN, then to national research network infrastructure including ERNET. as shown in Figure 5.3.

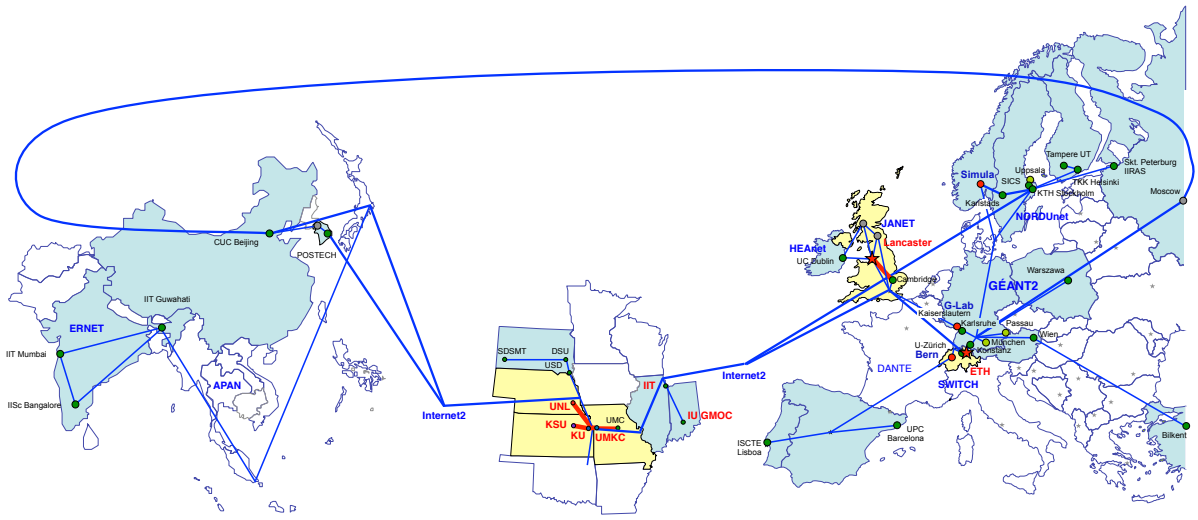

Figure 5.3: GpENI map

5.4.3 GpENI Deployment

The GpENI infrastructure [286] is in the process of expanding to 40 clusters with 200 nodes worldwide, federated with the larger GENI PlanetLab control framework and interconnected to several ProtoGENI facilities, as shown in Figure 5.3. This enables users to perform resilience and survivability experiments at at scale, both in terms of node count and with the geographic scope needed to emulate area-based challenges such as large-scale disasters. In our own research efforts, we are using these facilities to enable experiments that cross-verify the analytical and simulation-based resilience research currently underway at The University of Kansas [191], leveraging topology and challenge generation tools (KU-LoCGen and KU-CSM [193]) developed for this purpose, with emphasis on resilience metrics [219] and multi-path multi-realm diverse transport [211, 212] developed as part of our NSF FIND research in the PostModern Internet Architecture project [24].

5.5 Node Cluster Architecture

Each GpENI node cluster consists of several components, physically interconnected by a managed Netgear Gigabit-Ethernet switch to allow arbitrary and flexible experiments. GpENI uses a KanREN 198.248.240.0/21 IP address block within the `gpeni.net` domain; management access to the facility is via dual-homing of the Node Management and Experiment Control Processor. The node cluster is designed to be as flexible as possible at every layer of the protocol stack, and consists of the following components, as shown in Figure 5.4:

- GpENI management and control processor: general-purpose Linux machine
- PlanetLab control framework consisting of aggregate managers: MyPLC with GENI-wrapper SFA (at KSU), myVINI (at UMKC), and DCN (at UNL)
- PlanetLab programmable nodes (enabling layer 4 and 7 experimentation)
- VINI-based programmable routers (providing flexible network topologies), with Quagga and other extensions such as XORP and Click (enabling layer 3 experimentation), as well as the ability for GpENI partners to install their own programmable routers
- Site-specific experimental nodes and testbeds, including software defined radios (e.g. KUAR), optical communication laboratories, and sensor testbeds
- Managed Gigabit Ethernet switch, providing L2 VLAN programmability and connectivity to the rest of GENI
- Ciena optical switch running DCN providing L1 interconnection among GpENI node clusters on the Midwest US optical backbone

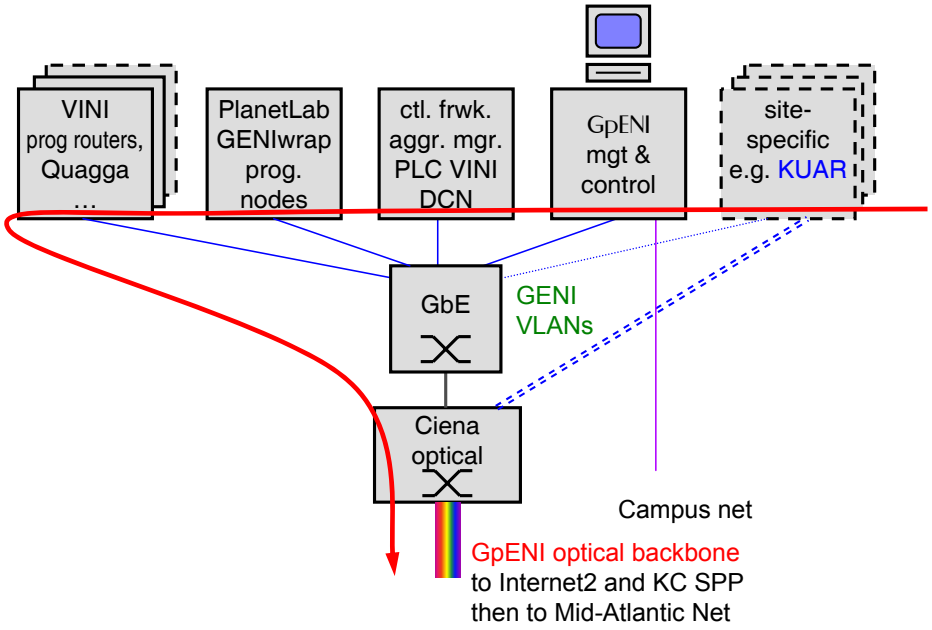

Figure 5.4: GpENI Node Cluster

5.5.1 GpENI Management and Control

The GpENI management and control services are distributed across the Linux machines dedicated for the purpose at each of the node clusters. Open-source tools are used wherever possible to minimise the amount of GpENI-specific software development and maintenance required. Some of these services are installed at every node cluster, for example the Cacti monitoring tool [297] is used to monitor the per-port network usage on each of the Netgear Gigabit-Ethernet switches. Nagios [298] is used to monitor the status of individual nodes and services across all the clusters. Zenoss Core [299] is also being evaluated as an alternative to Nagios. On the other hand, some functions are specific to one site, such as the GpENI-Planetlab demo [300], running on Apache and hosted only on the management node at the KSU node cluster.

The control node for each cluster also provides firewall and NAT services using Firestarter [301] for that cluster's private subnet, thereby protecting insecure devices, such as the Netgear

switch telnet and SNMP management interfaces, from direct exposure to the Internet.

5.5.2 Experiment Control

To ease experimenters role in resource discovery and preparation of experiments, the GENI User Shell (Gush) [302], provides a robust experiment control and management framework. Gush extends the PlanetLab control framework to implement an API that interacts with the GENI Clearinghouse. With Gush deployment on the GpENI aggregate, a user can reserve both PlanetLab and GpENI resources by downloading agents on the selected nodes and deploy their experiments subsequently with the help of the controller that communicates with the agent about node status. The Raven [303] provisioning service provides services such as the proper execution environment, software packages, configuration information and computational resources.

5.5.3 Methodology and Cross-Verification

Resilient topologies generated by KU-LoCGen and analyzed by KU-CSM are used to generate layer-2 topologies that configure the topology of GpENI experiments. We evaluate performance when slice topologies are challenged by correlated failures of nodes and links, measuring connectivity, packet delivery ratio, goodput, and delay, when subject to CBR, bulk data transfer, and transactional (HTTP) traffic. We also characterize the packet-loss probability of wireless links at the Utah Emulab, and the capabilities for emulating jamming and misbehaving nodes within the Emulab-federated CMU wireless emulator [295]. Workflow infrastructure is provided by Raven [303] to deploy experiments on these aggregates in an automated and repeatable manner.

5.5.4 Large-Scale Deployment

In order to provide the ability to experiment with non-IP network layers, the GENI federation has converged on ethernet VLANs as the common denominator across all testbeds. There are several resulting implications and technical challenges, for example, no matter the scale of the testbed (global in GpENI's case), it is one giant broadcast domain given the capabilities of commodity ethernet switches, and the usable L2 topology is restricted to a tree. The cost of native layer-2 interconnection on a global scale is also high. To address these challenges GpENI is deploying a number of emerging technologies both to manage the testbed itself as well as addressing the needs of experimenters. DCN (previously mentioned) is one such tool which establishes VLAN circuits across the testbed to manage broadcast traffic and provided a layer-2 point-to-point abstraction for experimenters. We have used L2TPv3 tunnels over IP research networks to mitigate the cost of long-distance layer-2 connectivity, however this still is limited to a tree topology. The *tinc* [304] project goes a step further, allowing the creation of a full mesh of VPN L2 tunnels while preventing broadcast storms and is a promising solution to these challenges.

Large scale resilience experiments are run over interconnected aggregates using DCN [292] (within GpENI) and OpenFlow and configured paths, with VINI/Planetlab layer-3 topologies, to emulate both existing ISP and synthetic topologies. Over these topologies we run our multipath-aware transport protocol ResTP to evaluate its performance under varying application and traffic loads. Based on the output of our challenge generation simulations, we selectively disable node slivers and links to emulate correlated network failures and attacks. In the future we will also use the wireless emulator under the ProtoGENI framework to emulate jamming attacks to wireless access networks. Each challenge set is classified as a single scenario, and each scenario is run multiple times to establish reasonable confidence in the results.

5.6 Summary

GpENI is an excellent testbed for the purpose of prototyping diversity protocols as well as cross-verifying simulation results, both because of its multi-layer programability which enables the crosslayering needed to expose divers paths to the end-to-end layer, as well as its worldwide scale. Due to the use of a standard Linux environment for experiments, are protocol prototypes are also able to run on other Linux-based platforms without rewriting code. The prototypes will continue to be developed by a number of students within the ResiliNets group who will be performing experiments based on them for their own thesis research.

Page left intentionally blank.

Chapter 6

Conclusions and Future Work

This dissertation presents the new resilience mechanism *path diversification* for enabling the use of end-to-end diversity to meet application service requirements in the face of various attacks and challenges. We find that no single mechanism can address all application requirements in all network conditions, and so propose the ResTP protocol, with a constrained set of operating modes that allow for appropriate tradeoffs to be chosen. This chapter presents conclusions drawn from the major contributions of the dissertation, and directions for future work.

6.1 Conclusions

Chapter 3 introduced *path diversification*, presenting its design and evaluation. We also discussed several metrics for evaluating path, node-pair, and graph diversity. We applied the path diversification mechanism to 17 real and synthetic networks and evaluated its ability to improve flow robustness in the presence of link and node failures. Path diversification provides a substantial performance improvement over conventional single-path mechanisms by using cross-layer information to make intelligent path selections based on the diversity. We then extended the applicability of the *total graph diversity* metric by compensating for topologies that have higher average hopcounts, thus creating

the cTGD metric. Our analysis of the properties of the topologies shows that cTGD is an excellent *predictor* of the survivability of these topologies when simultaneous distributed node and link failures occur.

Chapter 4 presents the ResTP and AeroTP protocols, ResTP being a multipath transport protocol designed for future Internet environments such as PoMo, and AeroTP being a domain specific transport protocol for aeronautical telemetry environments, that includes a subset of the ResTP functionality. In both cases a cross-layer architecture is used to provide information from the network layer with which to make intelligent flow-control and path selection decisions. Using ResTP we have shown a 20–30% performance improvement in the presence of link failures when diversity is available in the underlying network graph. Using AeroTP in a lossy environment we showed its significant benefits to performability by using opportunistic connection setup and decoupling loss from the flow-control mechanism. We also show the tradeoffs enabled by having both a fully-reliable and a quasi-reliable mode.

Chapter 5 discusses the development of the GpENI programmable testbed, and its applicability for developing prototype versions of ResTP and the ANTP suite, as well as cross-verifying these protocols with the simulation models. Due to its multi-layer programmability and worldwide scale it is particularly suited to these purposes. Experiments using ResTP and AeroTP are currently in preliminary phases and will continue in future work

6.2 Future Work

One improvement we would like to make to the path diversity evaluation is to base the failure scenarios on more detailed properties of the links in the network, using an annotated graph. For example a link depending on many optical amplifiers would be

assigned a greater probability of failure than a link with few or none. The next major evolution of this work centers around the move from random failure scenarios to realistic correlated failure models. These may be attack intelligently targeted at critical elements of the network architecture (high-degree nodes, or high-betweenness links for example). Then there are challenges of various types that affect geographic regions of the network [192, 193]. The ability to rigorously introduce and analyze such failures is essential in order to evaluate our *geographic path diversity* metric. Once a working classification of failure types and their effect on the network is established, the challenge simulation model [7] being developed by Egemen Çetinkaya can be used to evaluate the effectiveness of path diversification and ResTP against these types of challenges. Subsequently a challenge emulation framework will be incorporated into the GpENI testbed, to enable equivalent at-scale resilience tests of the ResTP implementation. With respect to the topology evaluation aspect of the work, we are looking at methods of generating a large set of topologies with specific properties (node degree, rank, betweenness, etc.) to use in further evaluating the accuracy of cTGD. We also want to develop algorithms for automatically generating topologies with high cTGD, subject to realistic cost constraints. It would also be valuable to develop an analytical proof of the properties of the cTGD metric.

We would like to make the multipath behavior of ResTP more dynamic, such that as paths fail additional paths could be requested from the path server (assuming a resilient path to the path server) or from a local cache of path candidates. This sort of on-demand path selection could limit the initial computational complexity while providing greater resilience to failed paths, and is something that merits further investigation. Additionally we plan to examine the effects of using closed and open-loop error correction methods simultaneously (i.e. hybrid ARQ) where ARQ is used to retransmit data that cannot be recovered through FEC. This presents an interesting optimization problem, because

while both error-correction mechanisms incur a cost in terms of overhead, ARQ also adds delay that will result in the hybrid scheme only being appropriate for applications that are tolerant of retransmission-induced delays. This frames the question then in terms of optimizing the overhead tradeoff between ARQ and FEC.

Bibliography

[1] James P. G. Sterbenz, Deep Medhi, Greg Monaco, Byrav Ramamurthy, Caterina Scoglio, Baek-Young Choi, Joseph B. Evans, Don Gruenbacher, Ronqing Hui, Wesley Kaplow, Gary Minden, and Jeff Verrant. Gpeni: Great plains environment for network innovation. `http://wiki.ittc.ku.edu/gpeni`, November 2009.

[2] James Ellis, David Fisher, Thomas Longstaff, Linda Pesante, and Richard Pethia. Report to the president's commission on critical infrastructure protection, 1997.

[3] David D. Clark. The design philosophy of the DARPA Internet protocols. In *Proceedings of the ACM SIGCOMM Symposium on Communications Architectures and Protocols*, pages 106–114, New York, NY, USA, 1988. ACM.

[4] Amund Kvalbein, Audun Fosselie Hansen, Tarik Cicic, Stein Gjessing, and Olav Lysne. Multiple routing configurations for fast IP network recovery. *IEEE Transactions on Networking*, 17(2):473–486, July 2008.

[5] Murtaza Motiwala, Megan Elmore, Nick Feamster, and Santosh Vempala. Path splicing. In *Proceedings of the ACM SIGCOMM conference on data communication*, pages 27–38, New York, NY, USA, August 17-22 2008. ACM.

[6] James P. G. Sterbenz and David Hutchison. Resilinets: Multilevel resilient and survivable networking initiative wiki. `http://wiki.ittc.ku.edu/resilinets`, April 2006.

[7] Egemen K. Çetinkaya, Dan Broyles, Amit Dandekar, Sripriya Srinivasan, and James P.G. Sterbenz. Modelling communication network challenges for future internet resilience, survivability, and disruption tolerance: A simulation-based approach. *Springer Telecommunication Systems*, 2011. (accepted March 2011).

[8] James P. G. Sterbenz, David Hutchison, Egemen K. Çetinkaya, Abdul Jabbar, Justin P. Rohrer, Marcus Schöller, and Paul Smith. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks: Special Issue on Resilient and Survivable Networks (COMNET)*, 54(8):1245–1265, June 2010.

[9] Algirdas Avizienis. Design of fault-tolerant computers. In *1967 Fall Joint Computer Conf.*, volume 31 of *AFIPS Conf. Proc.*, pages 733–743. Thompson Books, 1967.

[10] Algirdas Avizienis. Toward systematic design of fault-tolerant systems. *Computer*, 30(4):51–58, April 1997.

[11] T1A1.2 Working Group. Reliability-related metrics and terminology for network elements in evolving communications networks. American National Standard for Telecommunications T1.TR.524-2004, Alliance for Telecommunications Industry Solutions (ATIS), June 2004.

[12] P. A. Lee and T. Anderson. *Fault Tolerance: Principles and Practice*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1990.

[13] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Transactions on Dependable and Secure Computing*, 1(1):11–33, January-March 2004.

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

[15] R.J. Ellison, D. Fisher, R.C. Linger, H.F. Lipson, T. Longstaff, and N.R. Mead. Survivable network systems: An emerging discipline. Technical Report CMU/SEI-97-TR-013, Software Engineering Institute, Carnegie Mellon University, 1997.

[16] James P. G. Sterbenz, Rajesh Krishnan, Regina Rosales Hain, Alden W. Jackson, David Levin, Ram Ramanathan, and John Zao. Survivable mobile wireless networks: issues, challenges, and research directions. In *WiSE '02: Proceedings of the 3rd ACM workshop on Wireless security*, pages 31–40, New York, NY, USA, 2002. ACM Press.

[17] E. Jen. *Robust Design: A Repertoire of Biological, Ecological, and Engineering Case Studies*. Oxford University Press, 2005.

[18] John F. Meyer. Performability evaluation: Where it is and what lies ahead. In *Proceedings of the IEEE International Computer Performance and Dependability Symposium (IPDS)*, pages 334–343, April 1995.

[19] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[20] Hubert Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 28(4):425–432, Apr 1980.

[21] David D. Clark. Protocol design and performance. tutorial notes, IEEE INFOCOM, April 1995.

[22] James P. G. Sterbenz and Joseph D. Touch. *High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication*. Wiley, 1st edition, May 2001.

[23] Eytan Modiano. An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols. *Wireless Networks*, 5(4):279–286, 1999.

[24] Bobby Bhattacharjee, Ken Calvert, Jim Griffioen, Neil Spring, and James P. G. Sterbenz. Postmodern internetwork architecture. Technical Report ITTC-FY2006-TR-45030-01, Information and Telecommunication Center, 2335 Irving Hill Road, Lawrence, KS 66045-7612, February 2006.

[25] National science foundation website. `http://www.nsf.gov`, April 2008.

[26] NSF NeTS FIND initiative website. `http://www.nets-find.net`, April 2008.

[27] William Enck, Patrick Traynor, Patrick McDaniel, and Thomas La Porta. Exploiting open functionality in SMS-capable cellular networks. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 393–404, New York, NY, USA, 2005. ACM Press.

[28] Patrick Traynor, William Enck, Patrick McDaniel, and Thomas La Porta. Mitigating attacks on open functionality in SMS-capable cellular networks. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 182–193, New York, NY, USA, 2006. ACM Press.

[29] Bharat Bhargava, Xiaoxin Wu, Yi Lu, and Weichao Wang. Integrating heterogeneous wireless technologies: a cellular aided mobile ad hoc network (CAMA). *Mobile Networks and Applications*, 9(4):393–408, 2004.

[30] Al Hanbali. A survey of TCP over ad hoc networks. *IEEE Communications Surveys & Tutorials*, 7(3):22–36, 2005.

[31] W.A. Doeringer, D. Dykeman, M. Kaiserswerth, B.W. Meister, H. Rudin, and R. Williamson. A survey of light-weight transport protocols for high-speed networks. *Communications, IEEE Transactions on*, 38(11):2025–2039, Nov 1990.

[32] R Wang, T Taleb, A Jamalipour, and Bo Sun. Protocols for reliable data transport in space Internet. *IEEE Communications Surveys & Tutorials*, 11(2):21–32, 2009.

[33] Chonggang Wang, K Sohraby, Bo Li, M Daneshmand, and Yueming Hu. A survey of transport protocols for wireless sensor networks. *IEEE Network*, 20(3):34–40, 2006.

[34] A.J.D Rathnayaka, V.M Potdar, A Sharif, S Sarencheh, and S Kuruppu. Wireless sensor network transport protocol - a state of the art. In *Proceedings of the International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA)*, pages 812–817, 2010.

[35] Ka-Cheong Leung and V.O.K Li. Transmission control protocol (TCP) in wireless networks: Issues, approaches, and challenges. *IEEE Communications Surveys & Tutorials*, 8(4):64–79, 2006.

[36] V. Cerf, Y. Dalal, and C. Sunshine. Specification of Internet Transmission Control Program. RFC 675, December 1974.

[37] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168.

[38] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581 (Proposed Standard), April 1999. Obsoleted by RFC 5681, updated by RFC 3390.

[39] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), September 2001.

[40] Kevin Fall and Sally Floyd. Simulation-based comparisons of tahoe, reno and SACK TCP. *ACM SIGCOMM Computer Communication Review*, 26(3):5–21, July 1996.

[41] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. TCP Vegas: new techniques for congestion detection and avoidance. *SIGCOMM Comput. Commun. Rev.*, 24(4):24–35, 1994.

[42] J. G. Fletcher. Mechanisms for a reliable timer-based protocol. *Computer Networks*, pages 271–290, 1978.

[43] R. W. Watson. Timer-based mechanisms in reliable transport protocol connection management. *Computer Networks*, 5(1):47–56, February 1981.

[44] R. W. Watson and S. A. Mamrak. Gaining efficiency in transport services by appropriate design and implementation choices. *ACM Transactions on Computer Systems*, 5(2):97–120, May 1987.

[45] R. W. Watson. The Delta-T transport protocol: Features and experience. *Local Computer Networks*, 1989.

[46] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[47] A. Fraser. Towards a universal data transport system. *IEEE Journal on Selected Areas in Communications (JSAC)*, 1983.

[48] A.G Fraser and W.T Marshall. Data transport in a byte stream network. *IEEE Journal on Selected Areas in Communications (JSAC)*, 7(7):1020–1033, 1989.

[49] D. Velten, R.M. Hinden, and J. Sax. Reliable Data Protocol. RFC 908 (Experimental), July 1984. Updated by RFC 1151.

[50] C. Partridge and R.M. Hinden. Version 2 of the Reliable Data Protocol (RDP). RFC 1151 (Experimental), April 1990.

[51] A Baratz, J Gray, P. Jr Green, J Jaffe, and D Pozefsky. SNA networks of small systems. *IEEE Journal on Selected Areas in Communications*, 3(3):416–426, 1985.

[52] James Martin, Kathleen Kavanagh Chapman, and The Arben Group, CORPORATE Inc. *SNA: IBM's Networking Solution*. Prentice-Hall, Inc., Englewood Cliffs, NJ, April 1987.

[53] Thomas J. Routt. Distributed SNA: A network architecture gets on track. In *Network Systems Architecture*, pages 167–180. IEEE Press, Piscataway, NJ, USA, January 1992.

[54] D Cheriton and D Cheriton. VMTP: A transport protocol for the next generation of communication systems. *ACM SIGCOMM Computer Communication Review (CCR)*, 16(3):406–415, September 1986.

[55] D.R. Cheriton. VMTP: Versatile Message Transaction Protocol: Protocol specification. RFC 1045 (Experimental), February 1988.

[56] D.R Cheriton and C.L Williamson. VMTP as the transport layer for high-performance distributed systems. *IEEE Communications Magazine*, 27(6):37–44, 1989.

[57] David D. Clark, Mark L. Lambert, and Lixia Zhang. NETBLT: A high throughput transport protocol. *ACM SIGCOMM Computer Communication Review*, 17(5):353–359, October/November 1987.

[58] D.D. Clark, M.L. Lambert, and L. Zhang. NETBLT: A bulk data transfer protocol. RFC 969, December 1985. Obsoleted by RFC 998.

[59] D.D. Clark, M.L. Lambert, and L. Zhang. NETBLT: A bulk data transfer protocol. RFC 998 (Experimental), March 1987.

[60] M.L. Lambert. On testing the NETBLT Protocol over divers networks. RFC 1030, November 1987.

[61] K.K Ramakrishnan, Raj Jain, K.K Ramakrishnan, and Raj Jain. A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. In *Proceedings of the ACM SIGCOMM Symposium on Communications Architectures and Protocols*, Stanford, CA, August 1988. ACM.

[62] Van Jacobson. Congestion avoidance and control. *SIGCOMM Comput. Commun. Rev.*, 18(4):314–329, 1988.

[63] ITU-T X.214. Transport service definition for open systems interconnection for CCITT applications. ITU-T Recommendation X.214, November 1988.

[64] ITU-T X.224. Transport protocol specification for open systems interconnection for CCITT applications. ITU-T Recommendation X.224, November 1988.

[65] ITU-T X.234. Protocol for providing the OSI connectionless-mode tranposrt service. ITU-T Recommendation X.234, July 1994.

[66] R Jain. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *ACM SIGCOMM Computer Communication Review (CCR)*, 19(5):56–71, October 1989.

[67] James P.G. Sterbenz and Gurudatta M. Parulkar. AXON: A high speed communication architecture for distributed applications. In *Proceedings of the 9th Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*, pages 415–425, 1990.

[68] James P.G. Sterbenz and Gurudatta M. Parulkar. AXON: Application-oriented lightweight transport protocol design. In *Proceedings of the Tenth International Conference on Computer Communication (ICCC)*, pages 379–387, New Delhi, India, November 1990. Narosa Publishing House.

[69] Zheng Wang and Jon Crowcroft. A new congestion control scheme: Slow start and search (Tri-S). *ACM SIGCOMM Computer Communication Review (CCR)*, 21(1), January 1991.

[70] Zheng Wang and Jon Crowcroft. Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. *ACM SIGCOMM Computer Communication Review (CCR)*, 22(2):9–16, April 1992.

[71] Protocol Engines Inc. XTP protocol definition. Revision 3.6, XTP Forum, 1394 Greenworth Place, Santa Barbara CA 93108, January 1992.

[72] David C. Feldmeier. A framework of architectural concepts for high-speed communication systems. *IEEE Journal on Selected Areas in Communications*, 11(4):480–488, May 1993.

[73] Anthony J. McAuley. Weighted sum codes for error detection and their comparison with existing codes. *IEEE/ACM Trans. Netw.*, 2(1):16–22, 1994.

[74] S Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking (TON)*, 1(4):397–413, 1993.

[75] R. Braden. T/TCP – TCP Extensions for Transactions Functional Specification. RFC 1644 (Experimental), July 1994.

[76] A Bakre and B.R Badrinath. I-TCP: Indirect TCP for mobile hosts. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, pages 136–143, Vancouver, Canada, June 1995.

[77] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H Katz. Improving TCP/IP performance over wireless networks. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 2–11. ACM Press, 1995.

[78] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Proposed Standard), January 1996. Obsoleted by RFC 3550.

[79] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. Updated by RFC 5506.

[80] J C Lin and S Paul. RMTP: A reliable multicast transport protocol. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 1414–1424. IEEE Computer Society Press, 1996.

[81] Robert C. Durst, Gregory J. Miller, and Eric J. Travis. TCP extensions for space communications. In *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, pages 15–26, New York, NY, USA, November 1996. ACM Press.

[82] P. Danzig, Z. Liu, and L. Yan. An evaluation of TCP Vegas by live emulation, 1995.

[83] Janey C. Hoe. Improving the start-up behavior of a congestion control scheme for TCP. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 270–280, New York, NY, USA, 1996. ACM.

[84] Sally Floyd. TCP and explicit congestion notification. *SIGCOMM Comput. Commun. Rev.*, 24(5):8–23, 1994.

[85] Craig Partridge. *Gigabit Networking*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.

[86] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), October 1989. Updated by RFCs 1349, 4379.

[87] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001 (Proposed Standard), January 1997. Obsoleted by RFC 2581.

[88] V. Jacobson and R.T. Braden. TCP extensions for long-delay paths. RFC 1072, October 1988. Obsoleted by RFCs 1323, 2018.

[89] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. RFC 2018 (Proposed Standard), October 1996.

[90] V. Jacobson. Compressing TCP/IP Headers for Low-Speed Serial Links. RFC 1144 (Proposed Standard), February 1990.

[91] R. Fox. TCP big window and NAK options. RFC 1106, June 1989.

[92] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323 (Proposed Standard), May 1992.

[93] Kevin Brown and Suresh Singh. M-TCP: TCP for mobile cellular networks. *ACM SIGCOMM Computer Communication Review (CCR)*, 27(5):19–43, October 1997.

[94] ZJ Haas and P. Agrawal. Mobile-TCP: An asymmetric transport protocol design for mobile systems. In *Proceedings of the IEEE International Conference on Communications (ICC)*, volume 2, pages 1053–1058, Montreal, Canada, June 8–12 1997.

[95] K Chandran, S Ragbunathan, S Venkatesan, and R Prakash. A feedback based scheme for improving TCP performance in ad-hoc wireless networks. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS)*, pages 472–479, Amsterdam, Netherlands, May 1998.

[96] M. Allman, D. Glover, and L. Sanchez. Enhancing TCP Over Satellite Channels using Standard Mechanisms. RFC 2488 (Best Current Practice), January 1999.

[97] J.C. Mogul and S.E. Deering. Path MTU discovery. RFC 1191 (Draft Standard), November 1990.

[98] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the internet. *Networking, IEEE/ACM Transactions on*, 7(4):458–472, Aug 1999.

[99] M. Allman, S. Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, K. Scott, and J. Semke. Ongoing TCP Research Related to Satellites. RFC 2760 (Informational), February 2000.

[100] T. R. Henderson and R. H. Katz. Transport protocols for Internet-compatible satellite networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 17(2):326–344, 1999.

[101] J. Kulik, R. Coulter, D. Rockwell, and C. Partridge. Paced TCP for high delay-bandwidth networks. In *Proceedings of IEEE GLOBECOM*, December 1999.

[102] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960 (Proposed Standard), October 2000. Obsoleted by RFC 4960, updated by RFC 3309.

[103] T. Goff, J. Moronski, D.S. Phatak, and V. Gupta. Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environments. *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 3:1537–1545 vol.3, March 26–30 2000.

[104] Dongkyun Kim, C.-K Toh, and Yanghee Choi. TCP-BuS: Improving TCP performance in wireless ad hoc networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 1707–1713, 2000.

[105] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, and L. Vicisano. PGM Reliable Transport Protocol Specification. RFC 3208 (Experimental), December 2001.

[106] A Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 221–235, Rome, Italy, 2001.

[107] J Liu and S Singh. ATCP: TCP for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 19(7):1300–1315, 2001.

[108] Thomas D Dyer and Rajendra V Boppana. A comparison of TCP performance over three routing protocols for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 56–66, Long Beach, CA, 2001. ACM Press.

[109] I.F. Akyildiz, G. Morabito, and S. Palazzo. TCP-Peach: a new congestion control scheme for satellite ip networks. *Networking, IEEE/ACM Transactions on*, 9(3):307–321, Jun 2001.

[110] I.F Akyildiz, Xin Zhang, and Jian Fang. TCP-Peach+: Enhancement of TCP-Peach for satellite IP networks. *IEEE Communications Letters*, 6(7):303–305, 2002.

[111] Claudio Casetti, Mario Gerla, Saverio Mascolo, M. Y. Sanadidi, and Ren Wang. TCP Westwood: End-to-end congestion control for wired/wireless networks. *Wireless Networks*, 8(5):467–479, 2002.

[112] C.D.A Cordeiro, S.R Das, and D.P Agrawal. COPAS: Dynamic contention-balancing to enhance the performance of TCP over multi-hop wireless networks. In *Proceedings of the 11th International Conference on Computer Communications and Networks (IC3N)*, pages 382–387, Miami, FL, October 2002.

[113] S Kopparty, S.V Krishnamurthy, M Faloutsos, and S.K. Tripathi. Split TCP for mobile ad hoc networks. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 138–142, Taipei, Taiwan, November 2002.

[114] D Katabi and M Handley. Congestion control for high bandwidth-delay product networks. *Proceedings of the ACM SIGCOMM Conference*, 2002.

[115] Chieh-Yih Wan, Andrew T Campbell, and Lakshman Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 1–11, New York, NY, USA, 2002. ACM Press.

[116] C.-Y Wan, A.T Campbell, and L Krishnamurthy. Pump-slowly, fetch-quickly (PSFQ): A reliable transport protocol for sensor networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 23(4):862–872, April 2005.

[117] Prasun Sinha, Thyagarajan Nandagopal, Narayanan Venkitaraman, Raghupathy Sivakumar, and Vaduvur Bharghavan. WTCP: A reliable transport protocol for wireless wide-area networks. *Wireless Networks*, 8(2/3):301–316, 2002.

[118] Gavin Holland and Nitin Vaidya. Analysis of TCP performance over mobile ad hoc networks. *ACM Wireless Networks*, 8(2/3):275–288, 2002.

[119] Feng Wang and Yongguang Zhang. Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 217–225, Lausanne, Switzerland, 2002. ACM Press.

[120] A Lahanas and V Tsaoussidis. Improving TCP performance over networks with wireless components using "probing devices". In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 642–648, 2002.

[121] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649 (Experimental), December 2003.

[122] Tom Kelly. Scalable TCP: Improving performance in highspeed wide area networks. *ACM SIGCOMM Computer Communication Review (CCR)*, 33(2):83, April 2003.

[123] Cheng Peng Fu and S.C Liew. TCP Veno: TCP enhancement for transmission over wireless access networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 21(2):216–228, 2003.

[124] M Chinta, A Helal, and C Lee. ILC-TCP: An interlayer collaboration protocol for TCP performance improvement in mobile and wireless environments. *IEEE Wireless Communications and Networking (WCNC)*, 2:1004–1010, 2003.

[125] Eitan Altman and Tania Jimenez. Novel delayed ACK techniques for improving TCP performance in multihop wireless networks. In *Proceedings of Personal Wireless Communication*, pages 237–253, Venice, Italy, 2003. Springer.

[126] Z Fu, P Zerfos, H Luo, S Lu, L. Zhang, and M Gerla. The impact of multihop wireless channel on TCP throughput and loss. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1744–1753, 2003.

[127] Kaixin Xu, Mario Gerla, Lantao Qi, and Yantai Shu. Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 16–28, San Diego, CA, 2003. ACM Press.

[128] Luqing Yang, Winston K G Seah, and Qinghe Yin. Improving fairness among TCP flows crossing wireless ad hoc and wired networks. In *Proceedings of the the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 57–63, Annapolis, MD, 2003. ACM Press.

[129] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6):128–136, June 2003.

[130] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), November 2007.

[131] F Stann and J Heidemann. RMST: Reliable data transport in sensor networks. In *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, pages 102–112, 2003.

[132] Chieh-Yih Wan, Shane B Eisenman, and Andrew T Campbell. CODA: Congestion detection and avoidance in sensor networks. In *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 266–279, November 2003.

[133] Yogesh Sankarasubramaniam, Özgür B Akan, and Ian F Akyildiz. ESRT: Event-to-sink reliable trasport in wireless sensor networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, pages 177–188, New York, NY, USA, 2003. ACM Press.

[134] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The Secure Real-time Transport Protocol (SRTP). RFC 3711 (Proposed Standard), March 2004. Updated by RFC 5506.

[135] Kaiyu Zhou, K.L Yeung, and V.O.K Li. P-XCP: A transport layer protocol for satellite IP networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 2707–2711, 2004.

[136] O.B Akan, Jian Fang, and I.F Akyildiz. TP-Planet: A reliable transport protocol for interplanetary Internet. *IEEE Journal on Selected Areas in Communications (JSAC)*, 22(2):348–361, 2004.

[137] M Marchese, M Rossi, and G Morabito. PETRA: Performance enhancing transport architecture for satellite communications. *IEEE Journal on Selected Areas in Communications (JSAC)*, 22(2):320–332, 2004.

[138] Seung-Jong Park, Ramanuja Vedantham, Raghupathy Sivakumar, and Ian F Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 78–89, Japan, 2004.

[139] Consultative Comittee for Space Data Systems. CCSDS file delivery protocol (CFDP) part 1: Introduction and overview. Informational Report, Green Book CCDS 720.1-G-3, National Aeronauticas and Space Administration, Washington, DC, USA, April 2007.

[140] A Dunkels, T Voigt, and J Alonso. Distributed TCP caching for wireless sensor networks. In *Proceedings of the 3rd Annual Mediterranean Ad Hoc Networking Workshop*, Turkey, 2004.

[141] Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 134–147, New York, NY, USA, 2004. ACM Press.

[142] Cheng Tien Ee and Ruzena Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 148–161, New York, NY, USA, 2004. ACM Press.

[143] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In

Proceedings of the 1st Symposium on Network Systems Design and Implementation, pages 29–31, San Francisco, CA, USA, March 2004.

[144] Rajesh Krishnan, James P. G. Sterbenz, Wesley M. Eddy, Craig Partridge, and Mark Allman. Explicit transport error notification (ETEN) for error-prone wireless and satellite networks. *Comput. Netw.*, 46(3):343–362, 2004.

[145] Tarik Taleb, Nei Kato, and Yoshiaki Nemoto. An explicit and fair window adjustment method to enhance TCP efficiency and fairness over multihops satellite networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 22(2):371–387, 2004.

[146] Tarik Taleb, Nei Kato, and Yoshiaki Nemoto. A recursive, explicit and fair method to efficiently and fairly adjust TCP windows in satellite networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 4268–4274, 2004.

[147] Tarik Taleb, Nei Kato, and Yoshiaki Nemoto. REFWA: An efficient and fair congestion control scheme for LEO satellite networks. *IEEE/ACM Transactions on Networking (TON)*, 14(5):1031–1044, October 2006.

[148] Ming Zhang, Junwen Lai, Arvind Krishnamurthy, Larry Peterson, and Randolph Wang. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *Proceedings of the USENIX Annual Tchnical Conference*, Boston, MA, June 2004.

[149] S. Mascolo, L.A. Grieco, R. Ferorelli, P. Camarda, and G. Piscitelli. Performance evaluation of Westwood+ TCP congestion control. *Performance Evaluation*, 55(1-2):93–111, January 2004.

[150] Kai Xu, Ye Tian, and N Ansari. TCP-Jersey for wireless IP communications. *IEEE Journal on Selected Areas in Communications (JSAC)*, 22(4):747–756, 2004.

[151] D. Leith and R. Shorten. H-TCP: TCP for high-speed and long-distance networks. In *Proceedings of PFLDnet*, Argonne, 2004.

[152] E.H.-K Wu and Mei-Zhen Chen. JTCP: Jitter-based TCP for heterogeneous wireless networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 22(4):757–766, 2004.

[153] A.K Singh and K Kankipati. TCP-ADA: TCP with adaptive delayed acknowledgement for mobile ad hoc networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1685–1690, 2004.

[154] Hongwei Zhang, Anish Arora, Young-ri Choi, and Mohamed G Gouda. Reliable bursty convergecast in wireless sensor networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 266–276, May 2005.

[155] Chieh-Yih Wan, Shane B Eisenman, Andrew T Campbell, and Jon Crowcroft. Siphon: Overload traffic management using multi-radio virtual sinks in sensor networks. In *Proceedings of the 3rd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 116–129, San Diego, CA, USA, November 2005.

[156] Y.G Iyer, S Gandham, and S Venkatesan. STCP: A generic transport layer protocol for wireless sensor networks. In *Proceedings of the 14th IEEE International Conference on Computer Communications and Networks (ICCCN)*, pages 449–454, 2005.

[157] T Taleb, N Kato, and Y Nemoto. REFWA Plus: Enhancement of REFWA to combat link errors in LEO satellite networks. In *Proceedings of IEEE Workshop on High Performance Switching and Routing (HPSR)*, pages 172–176, 2005.

[158] M. E. Elaasar, M. Barbeau, E. Kranakis, and Zheyin Li. Satellite transport protocol handling bit corruption, handoff and limited connectivity. *IEEE Transactions on Aerospace and Electronic Systems*, 41(2):489–502, April 2005.

[159] Yangfan Zhou, M.R Lyu, Jiangchuan Liu, and Hui Wang. PORT: A price-oriented reliable transport protocol for wireless sensor networks. In *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering, (ISSRE)*, pages 117–126, Chicago, USA, 2005.

[160] S Biaz and N.H Vaidya. "de-randomizing" congestion losses to improve TCP performance over wired-wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 13(3):596–608, 2005.

[161] S Bhandarkar, N.E Sadry, A.L.N Reddy, and N.H Vaidya. TCP-DCR: A novel protocol for tolerating wireless channel errors. *IEEE Transactions on Mobile Computing*, 4(5):517–529, 2005.

[162] K Sundaresan, V Anantharaman, Hung-Yun Hsieh, and A R Sivakumar. ATP: A reliable transport protocol for ad hoc networks. *IEEE Transactions on Mobile Computing*, 4(6):588–603, 2005.

[163] H Shimonishi, M.Y Sanadidi, and M Gerla. Improving efficiency-friendliness tradeoffs of TCP in wired-wireless combined networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 3548–3552, 2005.

[164] H Shimonishi and T Hama. TCP-Adaptive Reno for improving efficiency-friendliness tradeoffs of TCP congestion control algorithm. *Proceedings of the*

International Workshop on Protocols for Future, Large-Scale & Diverse Network Transports (PFLDnet), 2006.

[165] V.C Gungor and O.B Akan. DST: Delay sensitive transport in wireless sensor networks. In *Proceedings of the IEEE 7th International Symposium on Computer Networks*, pages 116–122, Istanbul, Turkey, 2006.

[166] Chongang Wang, K Sohraby, V Lawrence, Bo Li, and Yueming Hu. Priority-based congestion control in wireless sensor networks. In *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pages 22–31, 2006.

[167] Shiwen Mao, D Bushmitch, S Narayanan, and S. S. Panwar. MRTP: A multiflow real-time transport protocol for ad hoc networks. *IEEE Transactions on Multimedia*, 8(2):356–369, 2006.

[168] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, and P. McDonald. When TCP breaks: Delay- and disruption- tolerant networking. *IEEE Internet Computing*, 10(4):72–78, July-Aug. 2006.

[169] Manikantan Ramadas, Scott Burleigh, and Stephen Farrell. Licklider transmission protocol–specification. Internet Draft, draft-irtf-dtnrg-ltp, Experimental, January 2008.

[170] S. Burleigh, M. Ramadas, and S. Farrell. Licklider Transmission Protocol - Motivation. RFC 5325 (Informational), September 2008.

[171] M. Ramadas, S. Burleigh, and S. Farrell. Licklider Transmission Protocol - Specification. RFC 5326 (Experimental), September 2008.

[172] S. Farrell, M. Ramadas, and S. Burleigh. Licklider Transmission Protocol - Security Extensions. RFC 5327 (Experimental), September 2008.

[173] Sukun Kim, Rodrigo Fonseca, Prabal Dutta, Arsalan Tavakoli, David Culler, Philip Levis, Scott Shenker, and Ion Stoica. Flush: A reliable bulk transport protocol for multihop wireless networks. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 351–365, Sydney, Australia, November 2007.

[174] Jeongyeup Paek and Ramesh Govindan. RCRT: Rate-controlled reliable transport for wireless sensor networks. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 305–319, Sydney, Australia, November 2007.

[175] Nurcan Tezcan and Wenye Wang. ART: An asymmetric and reliable transport mechanism for wireless sensor networks. *International Journal of Sensor Networks, Special Issue on Theoretical and Algorithmic Aspects in Sensor Networks*, 2:188–200, June 2007.

[176] B Marchi, A Grilo, and M Nunes. DTSN: Distributed transport for sensor networks. In *Proceedings of the 12th IEEE Symposium on Computers and Communications (ISCC)*, pages 165–172, Aveiro, Portugal, 2007.

[177] Muhammad Mostafa Monowar, Md. Obaidur Rahman, Al-Sakib Khan Pathan, and Choong Seon Hong. Congestion control protocol for wireless sensor networks handling prioritized heterogeneous traffic. In *Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, Dublin, Ireland, December 2008.

[178] E Giancoli, F Jabour, and A Pedroza. CTCP: Reliable transport control protocol for sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 493–498, Sydney, Australia, 2008.

[179] V.C Gungor, O.B Akan, and I.F Akyildiz. A real-time and reliable transport $(RT)^2$ protocol for wireless sensor and actor networks. *IEEE/ACM Transactions on Networking (TON)*, 16(2):359–370, 2008.

[180] C Miller and C Poellabauer. PALER: A reliable transport protocol for code distribution in large sensor networks. In *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 206–214, 2008.

[181] C Marcondes, M.Y Sanadidi, M Gerla, and H Shimonishi. TCP adaptive westwood, combining TCP westwood and adaptive reno: A safe congestion control proposal. In *IEEE International Conference on Communications (ICC)*, pages 5569–5575, 2008.

[182] S Liu, T Başar, and R Srikant. TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks. *ScienceDirect - Performance Evaluation*, 65:417–440, 2008.

[183] V Sharma, S Kalyanaraman, K Kar, K.K Ramakrishnan, and V Subramanian. MPLOT: A transport protocol exploiting multipath diversity using erasure codes. In *Proceedings of the 27th IEEE Conference on Computer Communications (INFOCOM)*, pages 121–125, 2008.

[184] Tuan Le, Wen Hu, Peter Corke, and Sanjay Jha. ERTP: Energy-efficient and reliable transport protocol for data streaming in wireless sensor networks. *Computer Communications*, 32(7-10):1154–1171, May 2009.

[185] MM Alam. CRRT: Congestion-aware and rate-controlled reliable transport in wireless sensor networks. *IEICE Transactions on Communications*, E92-B:184–189, January 2009.

[186] F.K Shaikh, A Khelil, A Ali, and N Suri. TRCCIT: Tunable reliability with congestion control for information transport in wireless sensor networks. In *Proceedings of the ICST Wireless Internet Conference (WICON)*, pages 1–9, Singapore, 2010.

[187] Peng Xie, J Yackoski, J.H Li, R Levy, and M Gerla. TCP-Derwood: A TCP variant to address link switches. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 644–649, 2010.

[188] Xia Zhuoqun, Chen Zhigang, Ye Hui, and Zhao Ming. An improved MPTCP in coded wireless mesh networks. In *Proceedings of the 2nd IEEE International Conference on Broadband Network & Multimedia Technology (IC-BNMT)*, pages 795–799, 2009.

[189] Rabat A. Mahmood. Simulating challenges to communication networks for evaluation of resilience. Master's thesis, The University of Kansas, Lawrence, KS, August 2009.

[190] Bijan Bassiri and Shahram Shah Heydari. Network survivability in large-scale regional failure scenarios. In *Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering (C3S2E)*, pages 83–87, New York, NY, USA, 2009. ACM.

[191] James P.G. Sterbenz, Egemen K. Çetinkaya, Mahmood A. Hameed, Abdul Jabbar, and Justin P. Rohrer. Modelling and analysis of network resilience (invited paper). In *Proceedings of the Third IEEE International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–10, Bangalore, India, January 2011.

[192] James P.G. Sterbenz, Egemen K. Çetinkaya, Mahmood A. Hameed, Abdul Jabbar, Qian Shi, and Justin P. Rohrer. Evaluation of network resilience, survivability, and disruption tolerance: Analysis, topology generation, simulation, and experimentation (invited paper). *Springer Telecommunication Systems*, 2011. (accepted March 2011).

[193] Egemen K. Çetinkaya, Dan Broyles, Amit Dandekar, Sripriya Srinivasan, and James P. G. Sterbenz. A comprehensive framework to simulate network attacks and challenges. In *Proceedings of the 2nd IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, pages 538–544, Moscow, Russia, October 18–20 2010.

[194] A. Sydney, C. Scoglio, M. Youssef, and P. Schumm. Characterising the robustness of complex networks. *International Journal of Internet Technology and Secured Transactions*, 2(3/4):291–320, December 2010.

[195] Wayne D. Grover and Demetrios Stamatelakis. Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network restoration. In *Proceeding of the IEEE International Conference on Communications (ICC'98)*, volume 1, pages 537–543, June 1998.

[196] L. Ford. Network flow theory. 1956.

[197] E. F. Moore. *The Shortest Path Through a Maze*. Bell Telephone System, 1959.

[198] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[199] Robert W. Floyd. Algorithm 97: Shortest path. *ACM Communications*, 5(6):345, 1962.

[200] Ramesh Bhandari. *Survivable Networks: Algorithms for Diverse Routing.* Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[201] MH MacGregor and WD Grover. Optimized k-shortest-paths algorithm for facility restoration. *Software: Practice and Experience*, 24(9), 1994.

[202] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4(2), 1974.

[203] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2), 1984.

[204] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker. In search of path diversity in ISP networks. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC '03)*, pages 313–318, New York, NY, USA, 2003. ACM.

[205] Xiaowei Yang and David Wetherall. Source selectable path diversity via routing deflections. *SIGCOMM Comput. Commun. Rev.*, 36(4):159–170, 2006.

[206] Johannes Lessmann, Marcus Schöller, and Frank Zdarsky. Rope ladder routing: Position-based multipath routing for wireless mesh networks. In *Proceedings of the IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6. IEEE Computer Society, June 2010.

[207] A. Atlas and A. Zinin. Basic Specification for IP Fast Reroute: Loop-Free Alternates. RFC 5286 (Proposed Standard), September 2008.

[208] N. Shillingford, D.C. Salyers, C. Poellabauer, and A. Striegel. DETOUR: Delay- and energy-aware multi-path routing in wireless ad hoc networks. In *Proceedings of the Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous)*, pages 1–8, August 2007.

[209] Yao Wang, Shivendra Panwar, Shunan Lin, and Shiwen Mao. Wireless video transport using path diversity: Multiple description vs layered coding. In *Proceedings of the International Conference on Image Processing*, volume 1, pages I–21–I–24, 2002.

[210] A.C. Begen, Y. Altunbasak, and O. Ergun. Multi-path selection for multiple description encoded video streaming. In *Proceedings of the IEEE International Conference on Communications (ICC '03)*, volume 3, pages 1583–1589, May 2003.

[211] Justin P. Rohrer, Ramya Naidu, and James P. G. Sterbenz. Multipath at the transport layer: An end-to-end resilience mechanism. In *Proceedings of the IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, pages 1–7, St. Petersburg, Russia, October 2009.

[212] Justin P. Rohrer, Abdul Jabbar, and James P. G. Sterbenz. Path diversification: A multipath resilience mechanism. In *Proceedings of the IEEE 7th International Workshop on the Design of Reliable Communication Networks (DRCN)*, pages 343–351, Washington, DC, USA, October 2009.

[213] Justin P. Rohrer and James P. G. Sterbenz. Predicting topology survivability using path diversity. In *Proceedings of the IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, pages 95–101, Budapest, Hungary, October 5–7 2011.

[214] Justin P. Rohrer, Abdul Jabbar, and James P. G. Sterbenz. Path diversification. *Springer Telecommunication Systems Journal*, invited 2011. (to appear).

[215] C. Cetinkaya and E.W. Knightly. Opportunistic traffic scheduling over multiple network paths. *INFOCOM*, 3:1928–1937, March 2004.

[216] Mark R. Carter, Mark P. Howard, Nicholas Owens, David Register, Jason Kennedy, Kelley Pecheux, and Aaron Newton. Effects of catastrophic events on transportation system management and operations, Howard Street tunnel fire, Baltimore City, Maryland – July 18, 2001. Technical report, U.S. Department of Transportation, ITS Joint Program Office, Washington DC, 2002.

[217] Hilary C. Styron. CSX tunnel fire: Baltimore, MD. US Fire Administration Technical Report USFA-TR-140, Federal Emergency Management Administration, Emmitsburg, MD, 2001.

[218] Resilinets topology map viewer. `http://www.ittc.ku.edu/resilinets/maps/`, January 2011.

[219] Abdul Jabbar Mohammad, David Hutchison, and James P. G. Sterbenz. Towards quantifying metrics for resilient and survivable networks. In *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP)*, pages 17–18, November 2006.

[220] Abdul Jabbar. *A Framework to Quantify Network Resilience and Survivability*. PhD thesis, The University of Kansas, Lawrence, KS, May 2010.

[221] Abdul Jabbar, Hemanth Narra, and James P. G. Sterbenz. An approach to quantifying resilience in mobile ad hoc networks. In *Proceedings of the 8th IEEE International Workshop on the Design of Reliable Communication Networks (DRCN)*, Krakow, Poland, October 2011.

[222] Rocketfuel: An ISP topology mapping engine, September 2008.

[223] KMI Corporation. North american fiberoptic long-haul routes planned and in place, 1999.

[224] Wikipedia: Degree distribution. `http://en.wikipedia.org/wiki/Degree_distribution`, May 2011.

[225] Ted G. Lewis. *Network Science*, chapter 2, page 24. John Wiley & Sons, Inc., 1st edition, 2009.

[226] M. E. J. Newman. *Networks: An Introduction*, chapter 7, page 199. Oxford University Press, 1st edition, 2010.

[227] Wikipedia: Clustering coefficient. `http://en.wikipedia.org/wiki/Clustering_coefficient`, May 2011.

[228] M. E. J. Newman. *Networks: An Introduction*, chapter 7, page 181. Oxford University Press, 1st edition, 2010.

[229] Priya Mahadevan, Dmitri Krioukov, Marina Fomenkov, Xenofontas Dimitropoulos, K. Claffy, and Amin Vahdat. The Internet AS-level topology: Three data sources and one definitive metric. *ACM Computer Communication Review*, 36(1):17–26, Jan. 2006.

[230] L. C. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40(1):35–41, March 1977.

[231] Justin P. Rohrer, Erik Perrins, and James P. G. Sterbenz. End-to-end disruption-tolerant transport protocol issues and design for airborne telemetry networks. In *Proceedings of the International Telemetering Conference*, San Diego, CA, October 27–30 2008.

[232] Justin P. Rohrer, Abdul Jabbar, Erik Perrins, and James P. G. Sterbenz. Cross-layer architectural framework for highly-mobile multihop airborne telemetry net-

works. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 1–9, San Diego, CA, USA, November 2008.

[233] Justin P. Rohrer and James P. G. Sterbenz. Performance and disruption tolerance of transport protocols for airborne telemetry networks. In *Proceedings of the International Telemetering Conference (ITC)*, Las Vegas, NV, October 2009.

[234] Kamakshi Sirisha Pathapati, Justin P. Rohrer, and James P. G. Sterbenz. Edge-to-edge ARQ: Transport-layer reliability for airborne telemetry networks. In *Proceedings of the International Telemetering Conference (ITC)*, San Diego, CA, October 2010.

[235] Justin P. Rohrer, Abdul Jabbar, Egemen K. Çetinkaya, and James P.G. Sterbenz. Airborne telemetry networks: Challenges and solutions in the ANTP suite. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 74–79, San Jose, CA, USA, November 2010.

[236] Kamakshi Sirisha Pathapati, Anh Nguyen, Justin P. Rohrer, and James P.G. Sterbenz. Performance analysis of the AeroTP transport protocol for highly-dynamic airborne telemetry networks. In *Proceedings of the International Telemetering Conference (ITC)*, Las Vegas, NV, October 2011.

[237] Justin P. Rohrer, Egemen K. Cetinkaya, Hemmanth Narra, Dan Broyles, Kevin Peters, and James P. G. Sterbenz. AeroRP performance in highly-dynamic airborne networks using 3D gauss-markov mobility model. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, Baltimore, MD, USA, November 7–10 2011.

[238] Justin P. Rohrer, Abdul Jabbar, Egemen K. Çetinkaya, Erik Perrins, and James P.G. Sterbenz. Highly-dynamic cross-layered aeronautical network architec-

ture. *IEEE Transactions on Aerospace and Electronic Systems (TAES)*, 47(4):2742–2765, October 2011.

[239] A. J. McAuley. Reliable Broadband Communication Using a Burst Erasure Correcting Code. *SIGCOMM Comput. Commun. Rev.*, 20(4):297–306, 1990.

[240] The network simulator: ns-2. `http://www.isi.edu/nsnam/ns/`, December 2007.

[241] Boston university representative internet topology generator (BRITE), September 2008.

[242] Global information grid (GIG) overarching policy. Department of Defense Directive, November 2003.

[243] Abdul Jabbar, Erik Perrins, and James P. G. Sterbenz. A cross-layered protocol architecture for highly-dynamic multihop airborne telemetry networks. In *Proceedings of the International Telemetering Conference (ITC)*, San Diego, CA, October 27–30 2008.

[244] iNET System Architecuture, version 2007.1. Central Test and Evaluation Investment Program (CTEIP), July 2007.

[245] iNET Needs Discernment Report, version 1.0. Central Test and Evaluation Investment Program (CTEIP), May 2004.

[246] iNET Technology Shortfalls Report, version 1.0. Central Test and Evaluation Investment Program (CTEIP), July 2004.

[247] iNET Working Group. http://www.inetprogram.org.

[248] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2(1):1–22, January 2004.

[249] L. Iannone, R. Khalili, K. Salamatian, and S. Fdida. Cross-layer routing in wireless mesh networks. *1st International Symposium on Wireless Communication Systems*, pages 319–323, 2004.

[250] J. Moy. OSPF Version 2. RFC 2328 (Standard), April 1998. Updated by RFC 5709.

[251] G. Malkin. RIP Version 2. RFC 2453 (Standard), November 1998. Updated by RFC 4822.

[252] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006.

[253] Mung Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, January 2007.

[254] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[255] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (Experimental), February 2007.

[256] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *The conference on Communications architectures, protocols and applications (SIGCOMM)*, pages 234–244, New York, NY, USA, 1994. ACM.

[257] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[258] Tyson Thedinger, Abdul Jabbar, and James P. G. Sterbenz. Store and haul with repeated controlled flooding. In *Second International IEEE Workshop on Mobile Computing and Networking Technologies (WMCNT)*, pages 728–733, Moscow, Russia, October 2010.

[259] Wen-Hwa Liao, Jang-Ping Sheu, and Yu-Chee Tseng. GRID: A fully location-aware routing protocol for mobile ad hoc networks. *Telecommunication Systems*, 18(1-3):37–60, 2001.

[260] M. G. de la Fuente and H. Ladiod. A performance comparison of position-based routing approaches for mobile ad hoc networks. *Vehicular Technology Conference (VTC)*, pages 1–5, October 2007.

[261] L. Galluccio, A. Leonardi, G. Morabito, and S. Palazzo. A MAC/routing cross-layer approach to geographic forwarding in wireless sensor networks. *Ad Hoc Netw.*, 5(6):872–884, 2007.

[262] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network*, 15(6):30–39, 2001.

[263] M. Yuksel, R. Pradhan, and S. Kalyanaraman. An implementation framework for trajectory-based routing in ad hoc networks. *Ad Hoc Networks*, 4(1):125–137, 2006.

[264] M. Iordanakis, D. Yannis, K. Karras, G. Bogdos, G. Dilintas, M. Amirfeiz, G. Colangelo, and S. Baiotti. Ad-hoc routing protocol for aeronautical mobile ad-hoc networks. In *Proceedings of the Fifth International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2006.

[265] Fabrice Tchakountio and Ram Ramanathan. Anticipatory routing for highly mobile endpoints. In *Proceedings of the Sixth IEEE Workshop on Mobile Computing*

Systems and Applications (WMCSA), pages 94–101, Washington, DC, USA, 2004. IEEE Computer Society.

[266] Hsin-Yuan Huang, A. Khendry, and T. G. Robertazzi. Layernet: a self-organizing protocol for small ad hoc networks. *IEEE Transactions on Aerospace and Electronic Systems*, 38(2):378–387, April 2002.

[267] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 3448 (Proposed Standard), January 2003. Obsoleted by RFC 5348.

[268] Egemen K. Çetinkaya and James P. G. Sterbenz. Aeronautical gateways: Supporting TCP/IP-based devices and applications over modern telemetry networks. In *Proceedings of the International Telemetering Conference*, Las Vegas, NV, October 26–29 2009.

[269] D. C. Feldmeier, A. J. McAuley, J. M. Smith, D. S. Bakin, W. S. Marcus, and T. M. Raleigh. Protocol boosters. *IEEE Journal on Selected Areas in Communications (JSAC)*, 16(3):437–444, April 1998.

[270] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. RFC 3135 (Informational), June 2001.

[271] David A. Maltz and Pravin Bhagwat. TCP Splice for application layer proxy performance. *Journal of High Speed Networks*, 8(3):225–240, 1999.

[272] Jean-Claude Laprie. Dependability: Basic concepts and terminology. Draft, IFIP Working Group 10.4 – Dependable Computing and Fault Tolerance, August 1994.

[273] iNET TmNS Ground Segment Architecture, version 2007. Central Test and Evaluation Investment Program (CTEIP), July 2007.

[274] iNET TmNS Test Article Segment Architecture, version 2007. Central Test and Evaluation Investment Program (CTEIP), July 2007.

[275] David Feldmeier. An overview of the TP++ transport protocol project. In Ahmed N. Tantawy, editor, *High Performance Networks: Frontiers and Experience*, volume 238 of *Kluwer International Series in Engineering and Computer Science*, chapter 8. Kluwer Academic Publishers, Boston, MA, USA, 1993.

[276] Kevin Fall and Stephen Farrell. DTN: An architectural retrospective. *IEEE Journal on Selected Areas in Communications (JSAC)*, 26(5):828–836, 2008.

[277] J. McQuillan, I. Richer, E. Rosen, B. Beranek, and N. Inc. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications*, 28(5):711–719, 1980.

[278] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers, and Robert Morris. Performance of multihop wireless networks: Shortest path is not enough. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, New Jersey, October 2002. ACM SIGCOMM.

[279] Simon Heimlicher, Merkourios Karaliopoulos, Hanoch Levy, and Thrasyvoulos Spyropoulos. On leveraging partial paths in partially-connected networks. In *Proceeding of the 28th IEEE Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brazil, April 2009.

[280] K. K. Ramakrishnan and Raj Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Transactions on Computer Systems*, 8(2):158–181, 1990.

[281] Md. Mamun-Or-Rashid, Muhammad Mahbub Alam, Md. Abdur Razzaque, and Choong Seon Hong. Congestion avoidance and fair event detection in wireless sensor network. *IEICE Transactions on Communications*, E90-B(12):3362–3372, 2007.

[282] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *The Journal of the Operational Research Society*, 49(3):237–252, March 1998.

[283] Cüneyt Özveren, Robert Simcoe, and George Varghese. Reliable and efficient hop-by-hop flow control. In *SIGCOMM '94: Proceedings of the conference on communications architectures, protocols and applications*, pages 89–100, New York, NY, USA, 1994. ACM.

[284] The ns-3 network simulator. `http://www.nsnam.org`, July 2009.

[285] Mohammed AL-Enazi, Santosh Ajith Gogi, Dongsheng Zhang, Egemen K. Çetinkaya, Justin P. Rohrer, and James P. G. Sterbenz. ANTP protocol suite software implementation architecture in python. In *International Telemetering Conference (ITC)*, Las Vegas, NV, October 2011.

[286] James P. G. Sterbenz, Deep Medhi, Byrav Ramamurthy, Caterina Scoglio, David Hutchison, Bernhard Plattner, Tricha Anjali, Andrew Scott, Cort Buffington, Gregory E. Monaco, Don Gruenbacher, Rick McMullen, Justin P. Rohrer, John Sherrell, Pragatheeswaran Angu, Ramkumar Cherukuri, Haiyang Qian, and Nidhi Tare. The Great plains Environment for Network Innovation (GpENI): A programmable testbed for future internet architecture research. In *Proceedings of the 6th International Conference on Testbeds and Research Infrastructures for the Development*

of Networks & Communities (TridentCom), pages 428–441, Berlin, Germany, May 18–20 2010.

[287] Justin P. Rohrer, Egemen K. Çetinkaya, and James P. G. Sterbenz. Progress and challenges in large-scale future internet experimentation using the GpENI programmable testbed. In *The 6th ACM International Conference on Future Internet Technologies (CFI)*, pages 46–49, Seoul, Korea, June 2011.

[288] Justin P. Rohrer, Egemen K. Çetinkaya, and James P.G. Sterbenz. Resilience experiments in the GpENI programmable future internet testbed. In *Proceedings of the 11th Würzburg Workshop on IP: Joint ITG and Euro-NF Workshop "Visions of Future Generation Networks" (EuroView2011)*, August 2011.

[289] ResumeNet. `http://www.resumenet.eu/project/index`, December 2009.

[290] The OpenFlow switch consortium. `http://www.openflowswitch.org/`, December 2009.

[291] G.J. Minden, J.B. Evans, L. Searl, D. DePardo, V.R. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill, A.M. Wyglinski, and A. Agah. KUAR: A flexible software-defined radio development platform. In *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pages 428–439, April 2007.

[292] Dynamic resource allocation via GMPLS optical network. `http://dragon.maxgigapop.net/`, November 2009.

[293] German lab. `http://www.german-lab.de/`, April 2010.

[294] PlanetLab. `http://www.planet-lab.org/`, November 2009.

[295] Emulab: Network emulation testbed. `http://www.emulab.net/`, December 2009.

[296] ProtoGENI wiki. http://www.protogeni.net/trac/protogeni, 2010.

[297] Cacti: the complete rrdtool-based graphing solution. http://www.cacti.net/, December 2009.

[298] Nagios. http://www.nagios.org/, December 2009.

[299] Zenoss: Unlegacy enterprise it management. http://www.zenoss.com/, December 2009.

[300] GpENI demonstration website. http://control-1.ksu.gpeni.net/demo/, November 2009.

[301] Firestarter: A modern linux firewall. http://www.fs-security.com/, December 2009.

[302] Gush: GENI user shell. http://gush.cs.williams.edu/trac/gush, December 2009.

[303] Raven provisioning service. http://raven.cs.arizona.edu/, December 2009.

[304] tinc wiki. http://www.tinc-vpn.org/, 2010.

Page left intentionally blank.

Appendix A

AeroTP Protocol Specification

This ANTP draft specification covers AeroTP: TCP-friendly disruption-tolerant aeronautical transport protocol for highly-mobile airborne networking in the iNET environment.

A.1 Architectural Overview

The iNET (Integrated Networked Enhanced Telemetry) program has identified a set of needs [NDR] for the T&E (test and evaluation) community that require a substantially enhanced networking capability for Major Range and Test Facility Bases. There is currently a significant effort underway in the iNET community to design the physical layer communications and MAC (medium access control), however a number of issues remain to be solved at the network and transport layers [TSR].

The current Internet protocols are unsuitable for the specific constraints and requirements of the aeronautical telemetry network environments in a number of respects [TSR]. At the same time, there is a need to be compatible with TCP/IP-based devices located on test articles (TAs) and relay nodes (RNs), as well as with the controlling applications at the ground station (GS).

The ANTP (Aeronautical Network and Transport Protocols for iNET) project is designing a new protocol suite for the upper layers of the emerging telemetry environment, shown in Figure A.1. These protocols are designed for the telemetry network environment, while fully interoperable with TCP/UDP/IP via gateways. The ANTP protocol suite consists of a domain-specific set of protocols appropriate for the highly-dynamic mobile-airborne telemetry environment. AeroTP is a disruption-tolerant TCP-friendly transport protocol with multiple reliability and QoS modes. AeroNP is an IP-compatible network protocol designed for efficient translation with IP. AeroRP is a highly-adaptive geolocation-assisted routing protocol. Translation between the ANTP and TCP/IP protocol stacks is performed at the AeroGW gateway.

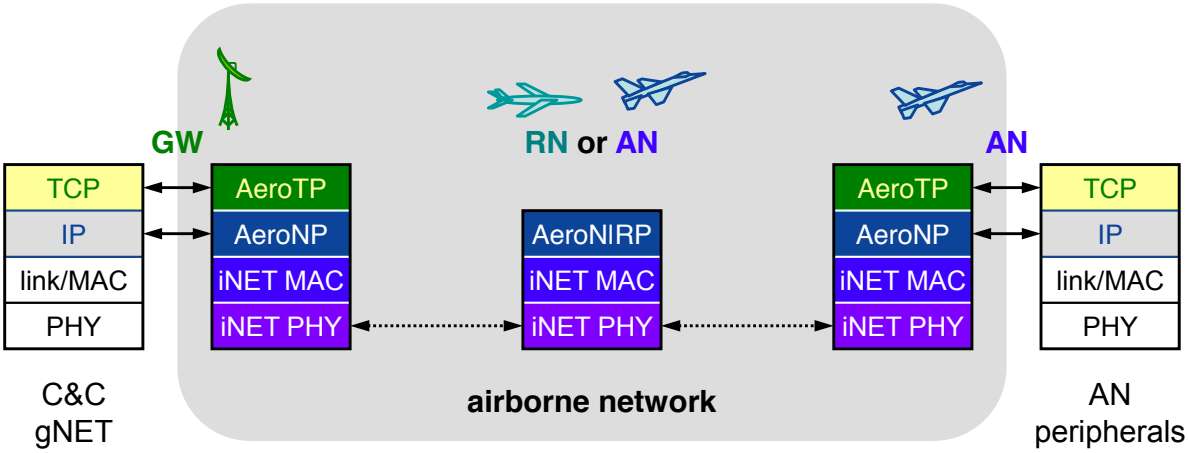

Figure A.1: Airborne network protocol architecture

A.2 Role of AeroTP

AeroTP is a new domain-specific transport protocol designed to meet the needs of the highly-dynamic network environment while being TCP-friendly to allow efficient splicing with conventional TCP at the AeroGWs in the GS and on the TA. Thus it transports TCP and UDP segments through the tactical network, but in an efficient manner that meets the needs of this environment: dynamic resource sharing, QoS support for fairness and

precedence, real-time data service, and bidirectional communication. AeroTP has several operational modes that support different reliability classes: reliable, quasi-reliable, best-effort connections, and best-effort datagrams. The first of these is fully TCP compatible, the last fully UDP compatible, and the others TCP-friendly with reliability semantics matching the needs of the mission and capabilities of the airborne network. The AeroTP header is designed to permit efficient translation between TCP/UDP and AeroTP by the AeroGW. AeroTP performs edge-to-edge data transfer between the edges of the TmNS and either terminates at native Aero devices or splices to TCP/UDP flows at the AeroGWs. Transport-layer functions that must be performed by AeroTP include connection setup and management, transmission control, and error control.

A.3 Conventions

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in RFC 2119 [RFC2119].

A.4 AeroTPDU Format

This section describes the format of the AeroTPDU (transport protocol data unit).

If a user data message does not fit into one AeroTPDU it can be fragmented into multiple chunks.

All integer fields in an AeroTPDU MUST be transmitted in network byte order, unless otherwise stated.

Source Port Number: 16 bits (unsigned integer)

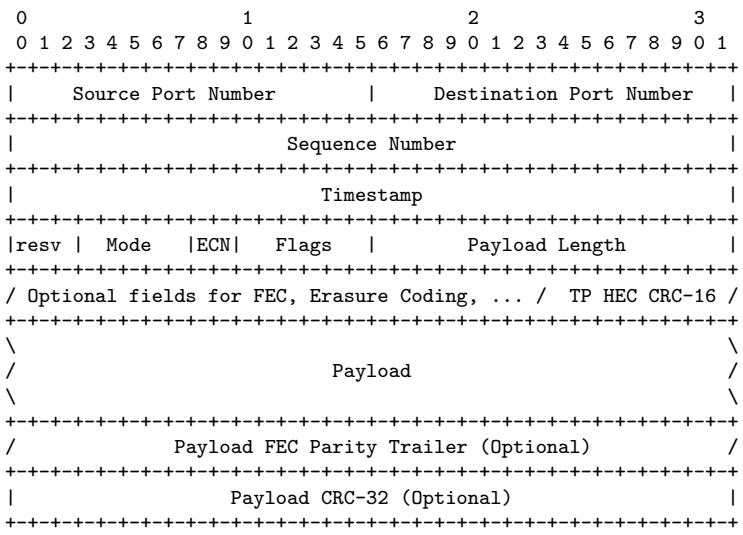

Figure A.2: AeroTP data segment structure

This is the AeroTP senders port number. It can be used by the receiver in combination with the source AeroNP address, the AeroTP destination port, and possibly the destination AeroNP address to identify the association (flow) to which this AeroTPDU belongs.

Destination Port Number: 16 bits (unsigned integer)

This is the AeroTP port number to which this AeroTPDU is destined. The receiving host will use this port number to de-multiplex the AeroTPDU to the correct receiving endpoint/application.

Sequence Number: 32 bits (unsigned integer)

This is the AeroTP sequence number assigned to this AeroTPDU. Connection-setup (ASYN) AeroTPDUs will always carry the sequence number 0. Subsequent AeroTPDUs are numbered using even numbers only. Odd sequence numbers are used only in the case that a previously transmitted AeroTPDU should be retransmitted using a smaller payload size. In this case the payload is split in half and retransmitted as two AeroTPDUs,

the first with the original sequence number, and the second using the original sequence number plus one. ACK piggybacking is for further study.

Timestamp: 32 bits (unsigned integer)

This is the time at which the AeroTPDU was last transmitted. The timestamp format is based on RFC-3339, neglecting punctuation characters. In decimal form from left to right, the first two digits represent the hour in the range 0023. The third and fourth digits represent the minute in the range 0059. The fifth and sixth digits represent the second in the range 0059. The remaining 4 digits represent the 4 highest-order decimal positions of the second, which have a range of 00009999. This results in a timestamp resolution of 0.1 ms, and a disambiguation period of 12 hrs.

Reserved: 3 bits reserved for future use.

Mode: 5 bits

This is the AeroTP operational mode currently in use, as defined as a particular set of mode bits, each of which represents an individual protocol option. The currently enumerated modes are Reliable (0x13), Quasi-Reliable (0x05), Unreliable Connection (0x01), and Unreliable Datagram (0x00). Any combination of the 5 mode bits is valid, however only the combinations enumerated in this document are recommended for use. The options represented by each bit are as follows: ARQ enable acknowledgement functions ERA enable erasure-coding functions FEC enable forward error correction CRC enable the payload cyclic redundancy check CON maintain connection state Several of these bits also indicate the use of associated optional fields.

ECN: 2 bits

This field contains unaltered TCP ECN flags used by AeroGWs for edge-to-edge transparency of TCP segments.

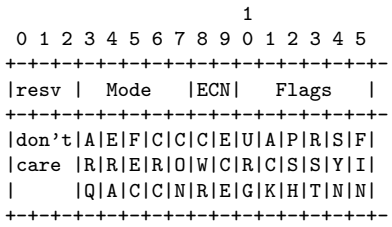

Figure A.3: AeroTP header options and flags

Flags: 6 bits

This contains AeroTP flags, some of which are used to pass unaltered TCP flags for translation to TCP segments. The individual AeroTP flags are as follows: URG used to pass the TCP urgent flag transparently ACK indicates this is an AACK, MACK, MNACK, ASYNACK, or AFINACK message PSH used to pass the TCP push flag transparently RST reset, use in AeroTP is a matter for further study SYN indicates an ASYN connection establishment request FIN indicates an AFIN connection termination request

Optional Fields: Variable length

Several of the mode bits indicate features that require additional header fields. If such bit is set in the mode field, the associated optional field must appear in the header. The size and use of such fields is a matter for further study, but preliminary values are below.

Error Correction (Optional): 4 bits

Indicates the strength of the Reed-Solomon forward error correction code if the FEC option is enabled.

Erasure Code (Optional): 4 bits

Indicates the block-erasure code algorithm used if the ERA option is enabled.

TP HEC CRC-16: 16 bits (unsigned integer)

This field contains the integrity check of this AeroTP header.

Payload:

Variable length data passed from application.

Payload CRC-32: 32 bits (unsigned integer)

This optional field at the end of the AeroTPDU contains the integrity check of the payload if the CRC mode bit is set.

A.5 Multiple ACK (MACK) packet format

This section describes the format of the MACK (Multiple Acknowledgement) used for acknowledging receipt of multiple packets.

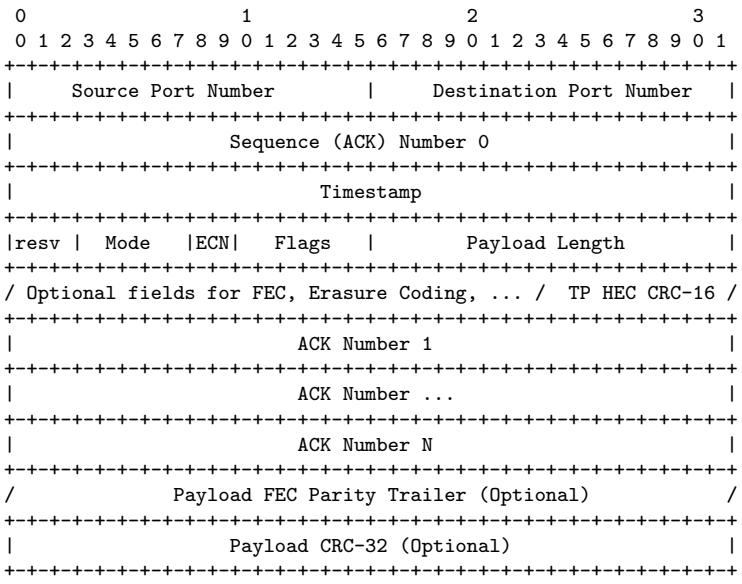

Figure A.4: AeroTP MACK segment structure

Field definitions are identical to those used in the previous section.

ACK Number X: 32 bits (unsigned integer)

These are the 32-bit sequence numbers of each packet being acknowledged.

A.6 AeroTP Flow Management

Connection and flow management vary depending on the operational mode in use. This section gives an overview of the state transitions used in a generalized case.

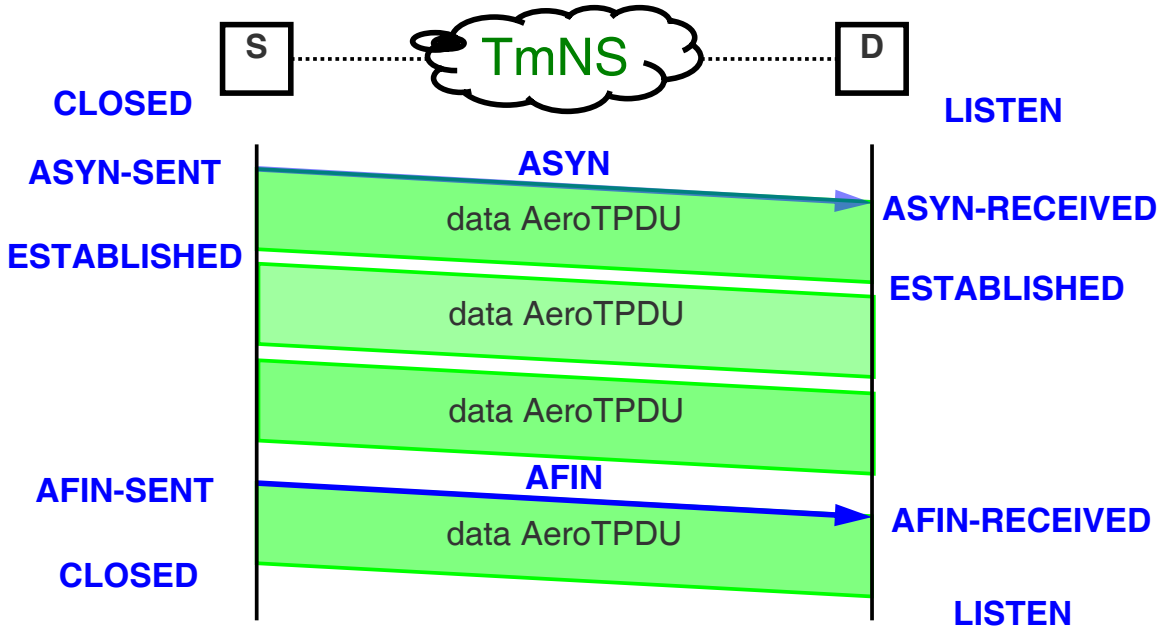

Figure A.5: AeroTP connection management

A.6.1 Connection Establishment

For connection-oriented modes (reliable, quasi-reliable, and unreliable connection-oriented), the AeroTP connection is initiated by calling the CONNECT method. This sets the SYN flag in the header of the first AeroTPDU in the flow for the ASYN connection setup message (sequence number = 0) and transmits it. If there is data in the send buffer, it will be inserted in the payload of the first AeroTPDU, otherwise the payload will be empty. When the ASYN is received, the receiver enters the SYN-RECEIVED state. When the

first data AeroTPDU is received (which may or may not be in the ASYN message) is received, the receiver enters the ESTABLISHED state. In reliable mode, connection setup is acknowledged by an ASYNACK message indicated by setting the SYN and ACK flags in the header of the return AeroTPDU (sequence number = 0). When the ASYNACK is received, the sender enters the ESTABLISHED state.

A.6.2 Connection Termination

The AeroTP connection is terminated with an AFIN message, by setting the FIN flag in the header of the last data AeroTPDU in the flow, or by sending an AeroTPDU with no payload and the FIN flag set. When the AFIN has been transmitted, the sender enters the AFIN-SENT state. When the AFIN is received, the receiver enters the AFIN-RECEIVED state and when it finishes processing the AFIN AeroTPDU it enters the LISTEN state. In fully-reliable mode and nearly-reliable mode, connection termination is acknowledged by setting the FIN and ACK flags on either the last data TPDU or by generating a separate AFINACK message with an empty payload. When the AFINACK is received, the sender enters the CLOSED state.

Connectionless Operation

AeroTP may operate in a stateless mode (unreliable connectionless) in which case no ASYN, ASYNACK, MACK, AFIN, or FINACK messages are used.

A.7 AeroTP Data Transfer

AeroTP uses one of several reliability modes to transfer data, depending on the class of service required. Each mode uses several mechanisms to optimize its performance. This section details the behavior of each mode and the mechanisms it uses.

A.7.1 Reliable Mode

Reliable mode is connection-oriented, and uses positive acknowledgements in combination with the payload CRC to confirm correct data delivery.

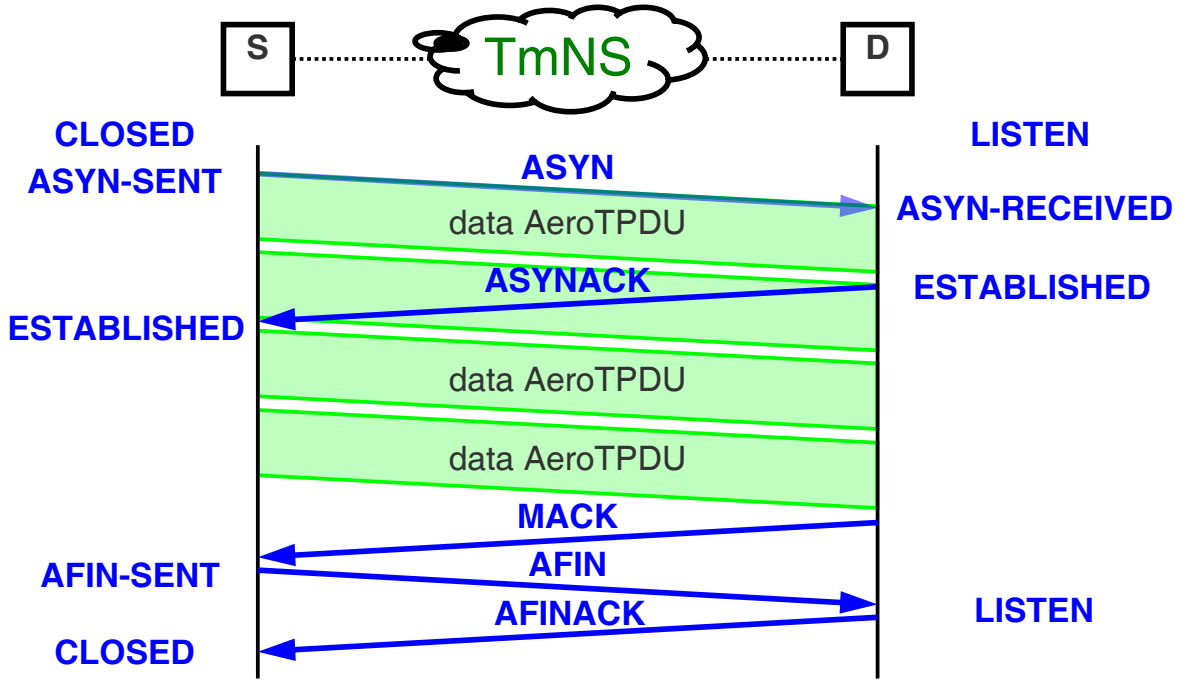

Figure A.6: AeroTP compact reliable mode

In order to reduce latency, the application may begin data transmission in the same AeroTPDU as the SYN flag is sent.

Alternatively, the application may wait until the connection reaches the established state to begin transmitting data segments. The AeroTP reliable mode is designed to preserve end-to-end reliability. To accomplish this, data segments are not acknowledged until the receiving application (or gateway) approves them. It is up to the application whether to verify data before acknowledging it.

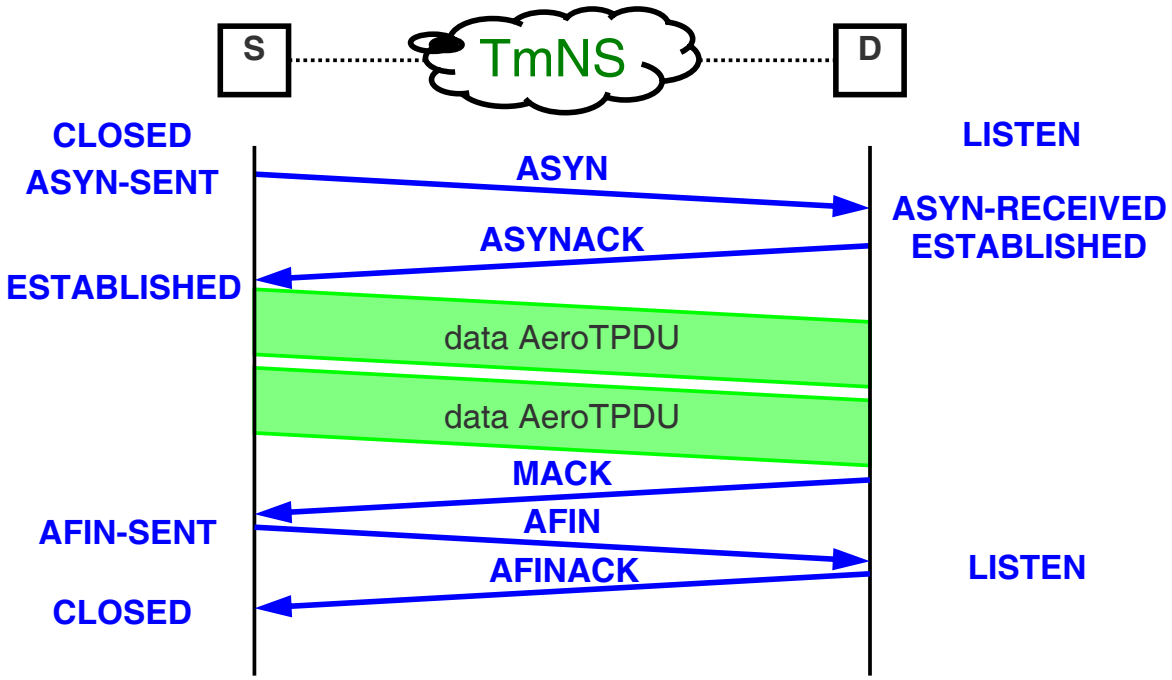

Figure A.7: AeroTP reliable mode

A.7.2 Quasi-Reliable Mode

Quasi-reliable mode is connection oriented, and uses forward-error-correction (FEC) to achieve statistical reliability.

A.7.3 Unreliable Connection-Oriented Mode

Unreliable connection-oriented mode may optionally use the payload CRC to check for errors, but provides no other assurance of reliable delivery.

A.7.4 Unreliable Connectionless Mode

Unreliable connectionless mode provides no assurance of reliable delivery or data correctness.

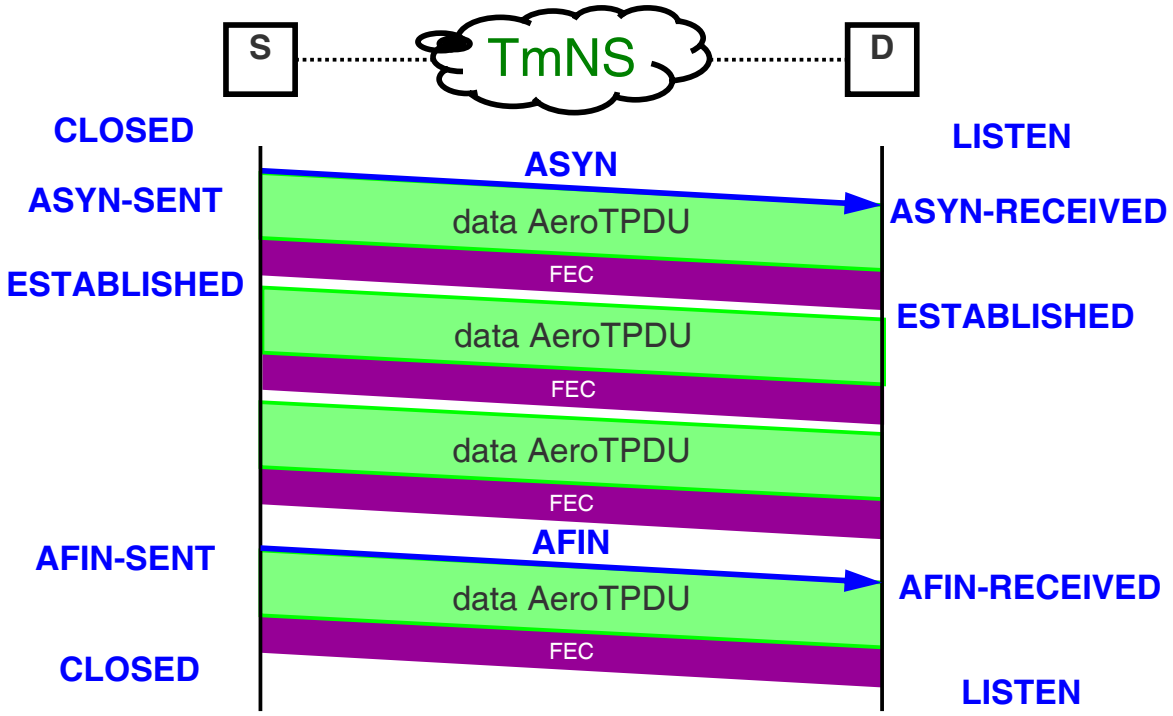

Figure A.8: AeroTP quasi-reliable mode

A.7.5 Security Considerations

Impact of iNET security architecture on AeroTP is expected to be minimal. The contents of the AeroTP payload may be encrypted, and the AeroTPDU itself may be encrypted for encapsulation in a lower-layer frame without any adverse effects on the operation of AeroTP. To the extent that lower-layer statistics on bit and/or packet error rates (BER and PER) are encrypted, AeroTPs ability to adapt to channel conditions may be impaired, however by recording statistics on errors and packet loss at the transport layer it will be partially self-sufficient in this respect. This is discussed in the ANTP cross-layering specification [AeroXL]. Initial sequence number negotiation is for further study.

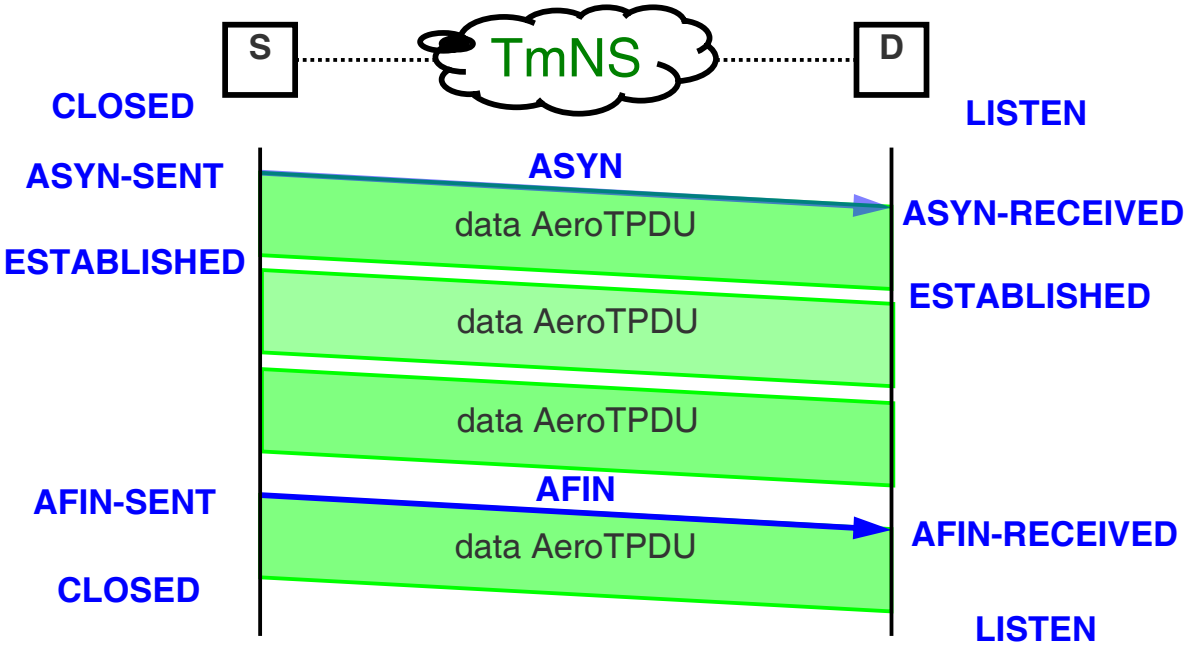

Figure A.9: AeroTP unreliable connection-oriented mode

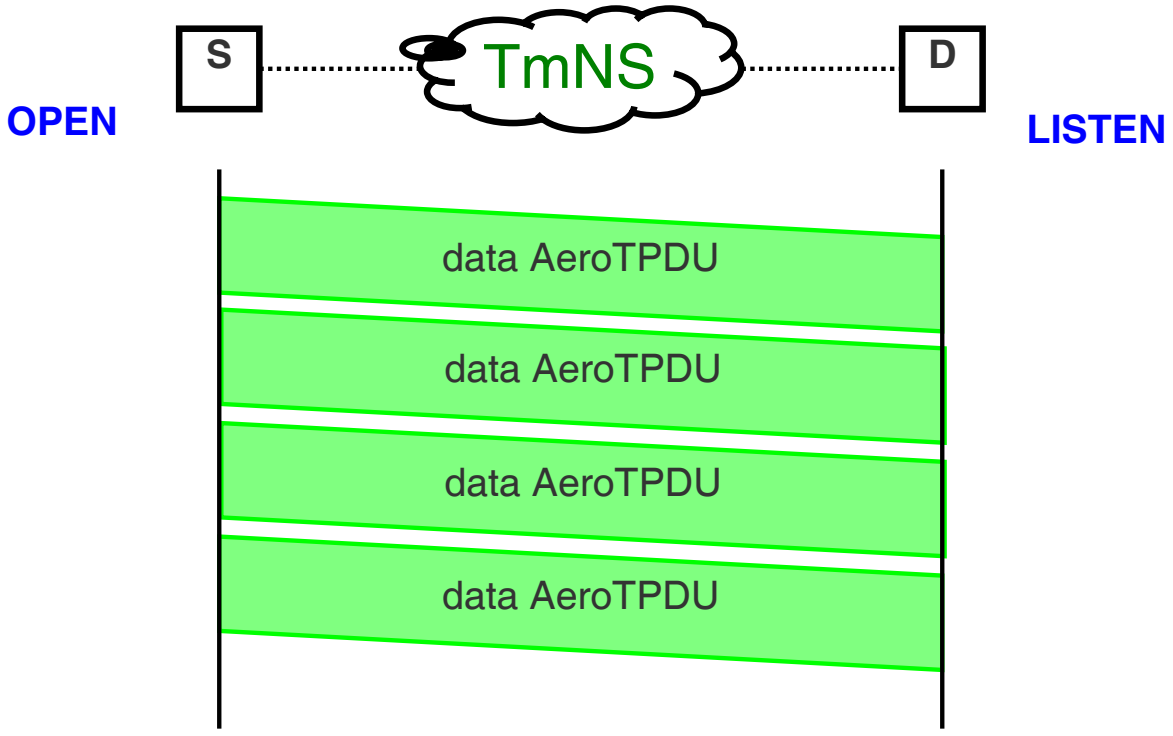

Figure A.10: AeroTP unreliable mode

Page left intentionally blank.

Appendix B

Tabular Data

Table B.1: Network Characteristics

Network	**Nodes**	**Links**	**Avg. Node Degree**	**TGD** $k = 4$	**Clustering Coefficient**	**Diam.**	**Radius**	**Hopcount**	**Closeness**	**Node Between.**	**Link Between.**
Full-Mesh	20	190	**19.00**	**0.9502**	**1**	**1**	**1**	**1**	**1**	**0**	**1**
Manhattan Grid	25	40	3.20	0.8964	0	8	4	3.3333	0.3067	**110**	54
Ring	25	25	2.00	0.6321	0	12	12	6.5	0.1538	132	78
Star	25	24	1.92	0.000	0	**2**	**1**	1.92	0.5302	552	24
AboveNet	22	80	**7.27**	0.8559	**0.6514**	**3**	**2**	**1.7229**	**0.5947**	196	**21**
AT&T	108	141	2.61	0.5881	0.3274	6	3	3.3790	0.3030	4160	943
AT&T Phys.	361	466	2.58	**0.9014**	0.0550	37	19	13.57	0.0763	4527	1893
EBONE	28	66	4.71	0.8635	0.3124	4	3	2.2804	0.4507	132	42
Exodus	22	51	4.64	0.8843	0.3307	4	**2**	2.0563	0.4978	132	22
GÉANT2 Phys.	34	51	3.00	0.7623	0.2898	9	5	3.4652	0.3007	556	131
Level 3	53	456	**17.20**	**0.9154**	**0.7333**	4	**2**	**1.7721**	**0.5845**	664	84
Sprint	44	106	4.82	0.8120	0.3963	5	3	2.6882	0.3853	602	129
Sprint Phys.	263	311	2.37	0.8821	0.0340	37	19	14.78	0.0700	3609	1637
Telstra	58	60	2.07	0.1295	0.2411	6	3	3.3025	0.3095	2136	806
Tiscali	51	129	5.06	0.7785	0.5068	5	3	2.4298	0.4236	656	96
Verio	122	310	5.08	0.8104	0.3509	8	4	3.1026	0.3335	3736	480
VSNL	7	7	2.00	0.2001	0.4167	4	**2**	2.0952	0.4982	**18**	**12**

Table B.2: Network Rank

Network	**Survivability Rank**	**Node Deg. Rank**	**TGD Rank**	**Clustering Rank**	**Diam. Rank**	**Hopcount Rank**	**Closeness Rank**	**Node Bet. Rank**	**Link Bet. Rank**
Full-Mesh	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Level 3	**2**	**2**	**2**	**2**	4	**2**	**3**	10	9
AboveNet	**3**	**3**	8	**3**	**3**	**3**	**2**	5	**3**
Exodus	4	5	5	8	4	5	6	4	4
EBONE	5	5	7	10	4	7	7	4	6
Tiscali	6	4	11	4	5	8	8	9	10
Sprint	7	5	9	6	5	9	9	8	11
Verio	8	4	10	7	7	10	10	13	13
Manhattan Grid	9	6	4	15	7	12	12	**3**	7
VSNL	10	7	15	5	4	6	5	**2**	**2**
GÉANT2 Phys.	11	6	12	11	8	14	14	7	12
Star	12	8	17	15	**2**	4	4	6	5
AT&T	13	7	14	9	6	13	13	14	15
Telstra	14	7	16	12	6	11	11	11	14
Ring	15	7	13	15	9	15	15	4	8
AT&T Phys.	16	7	**3**	13	10	16	16	15	17
Sprint Phys.	17	7	6	14	10	17	17	12	16

Table B.3: Compensated TGD

Network	**Survivability Rank**	**cTGD**	**cTGD Rank**
Full-Mesh	**1**	**0.9514**	**1**
Level 3	**2**	**0.4494**	**2**
AboveNet	**3**	**0.4386**	**3**
Exodus	4	0.3617	4
EBONE	5	0.3113	5
Tiscali	6	0.2641	6
Sprint	7	0.2407	7
Verio	8	0.2009	8
Manhattan Grid	9	0.2002	9
VSNL	10	0.1783	10
GÉANT2 Phys.	11	0.1668	11
Star	12	0.1628	12
AT&T	13	0.1446	13
Telstra	14	0.0941	14
Ring	15	0.0667	15
AT&T Phys.	16	0.0348	16
Sprint Phys.	17	0.0307	17

Appendix C

Maps

This appendix contains a set of maps for the topologies used in the analysis of the *path diversification* metrics. The maps are produced by taking screenshots of the KU-TopView web-based tool located at `http://www.ittc.ku.edu/resilinets/maps/`. To recreate the maps, use `#990000` for the link color, `#999999` for the primary marker color, and `#000000` for the marker corner color.

C.1 Physical Topologies

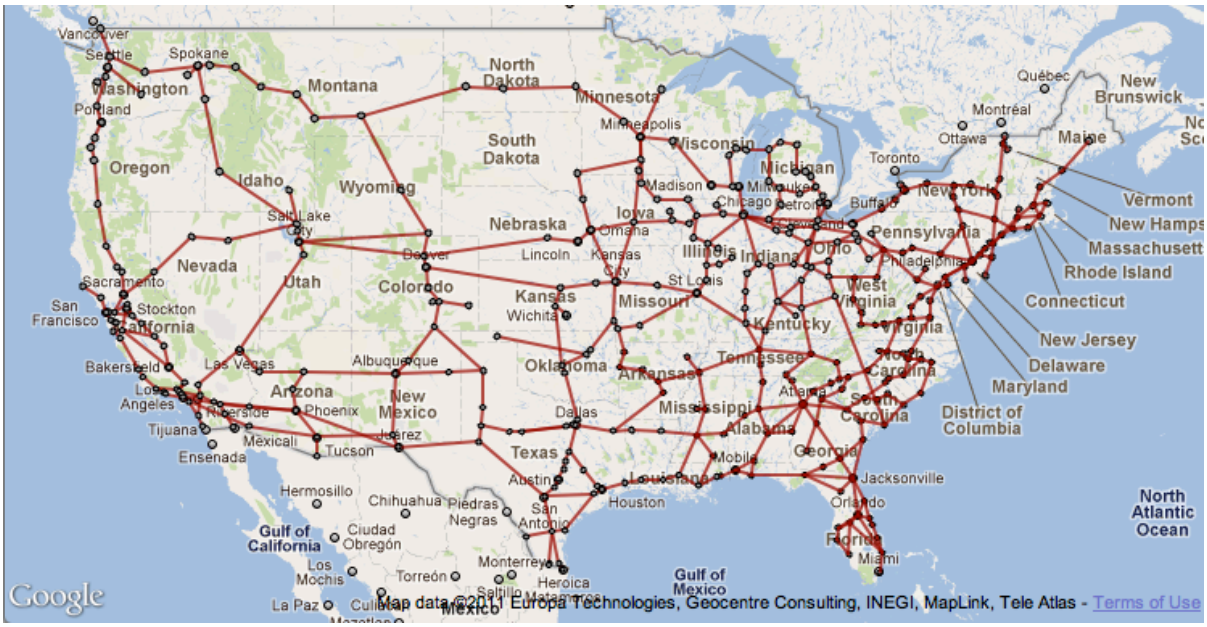

Figure C.1: AT&T physical topology

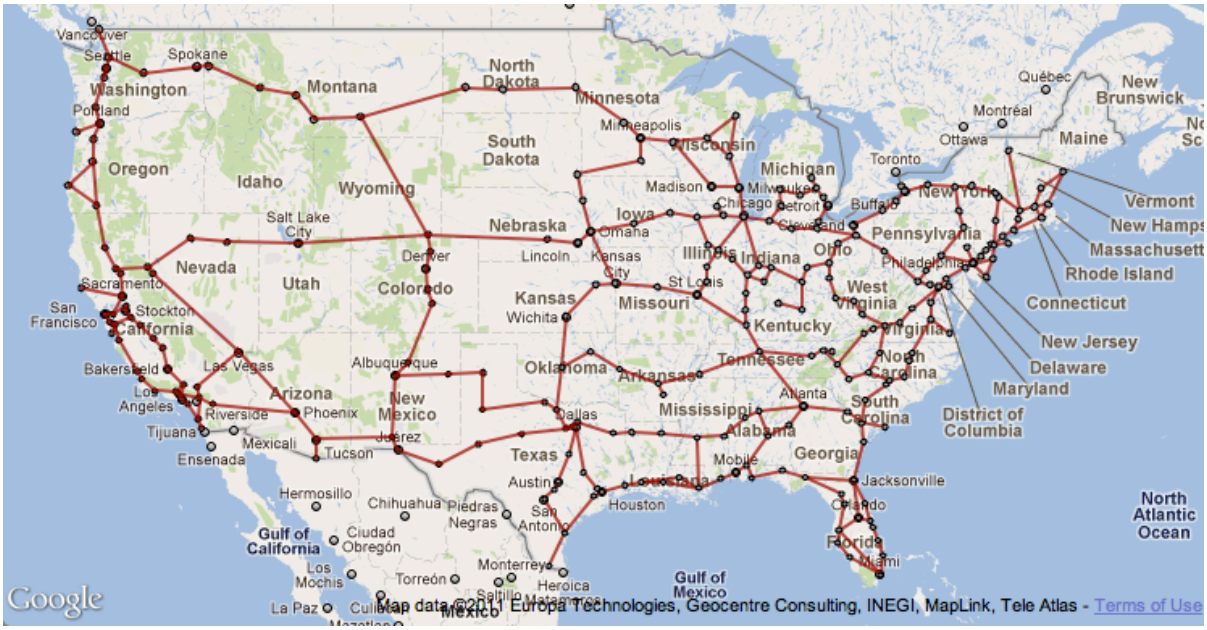

Figure C.2: Sprint physical topology

C.2 Logical Topologies

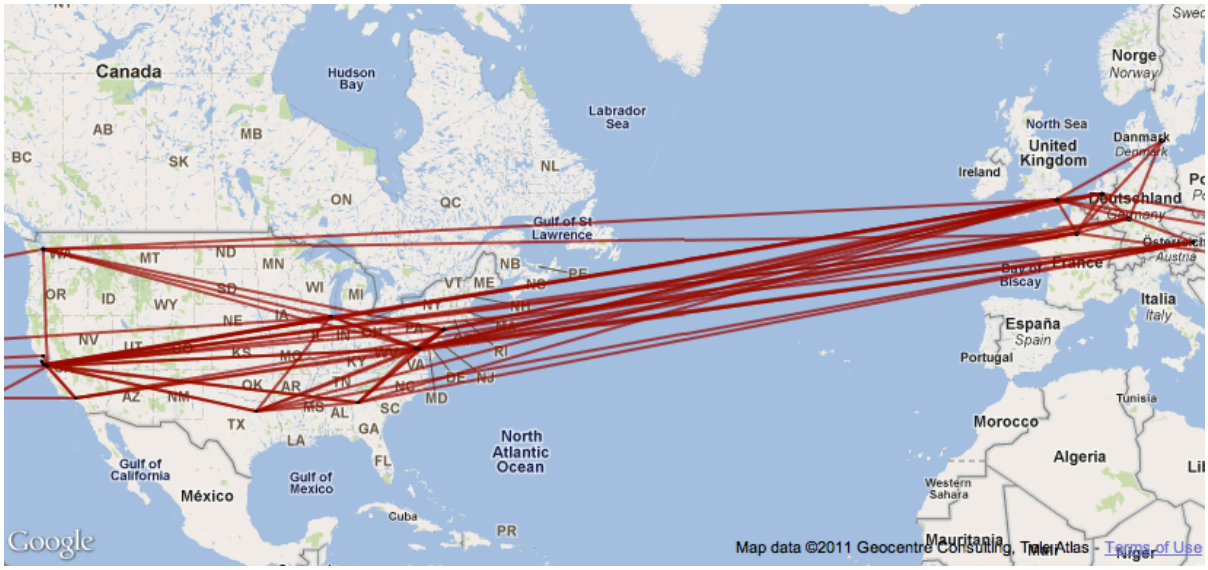

Figure C.3: AboveNet logical topology

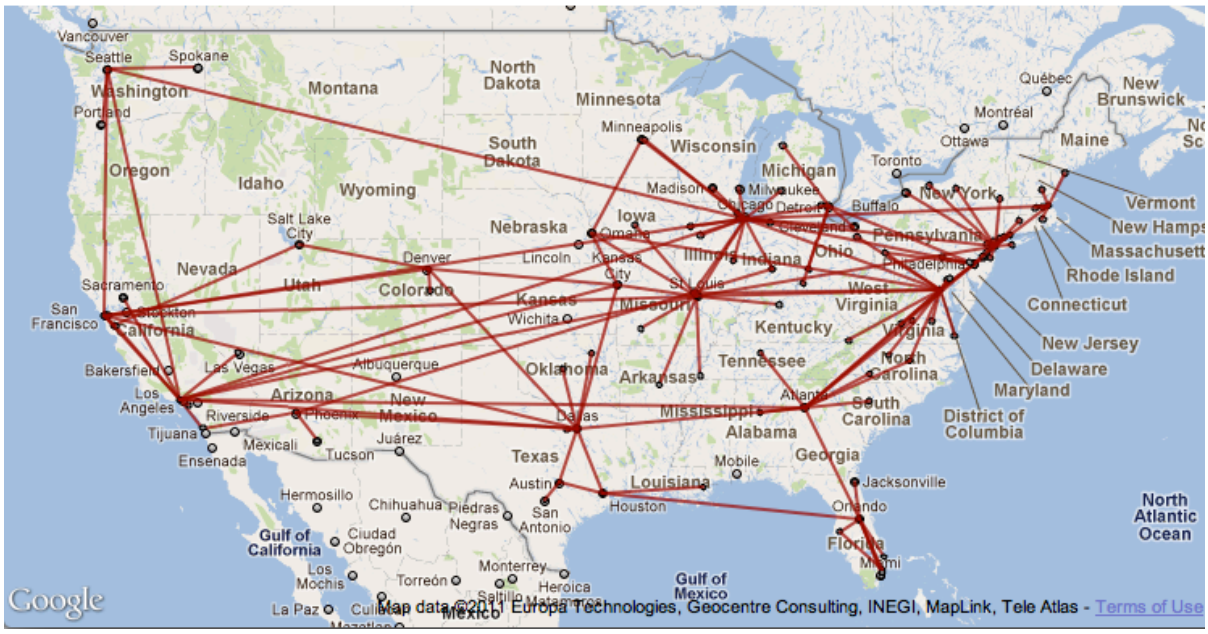

Figure C.4: AT&T logical topology

Figure C.5: EBONE logical topology

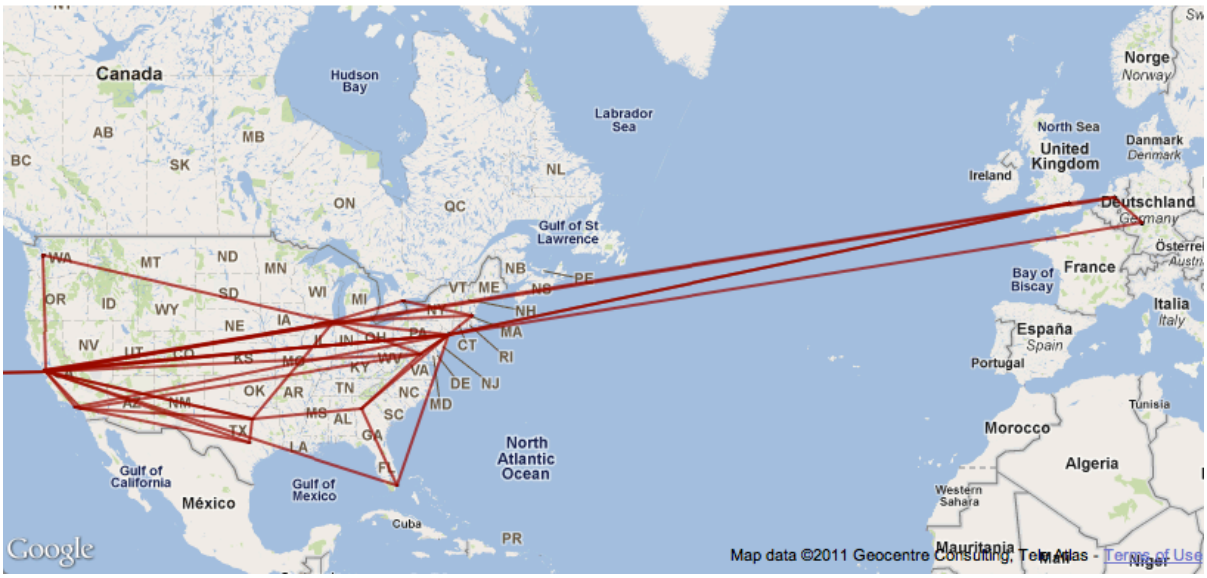

Figure C.6: Exodus logical topology

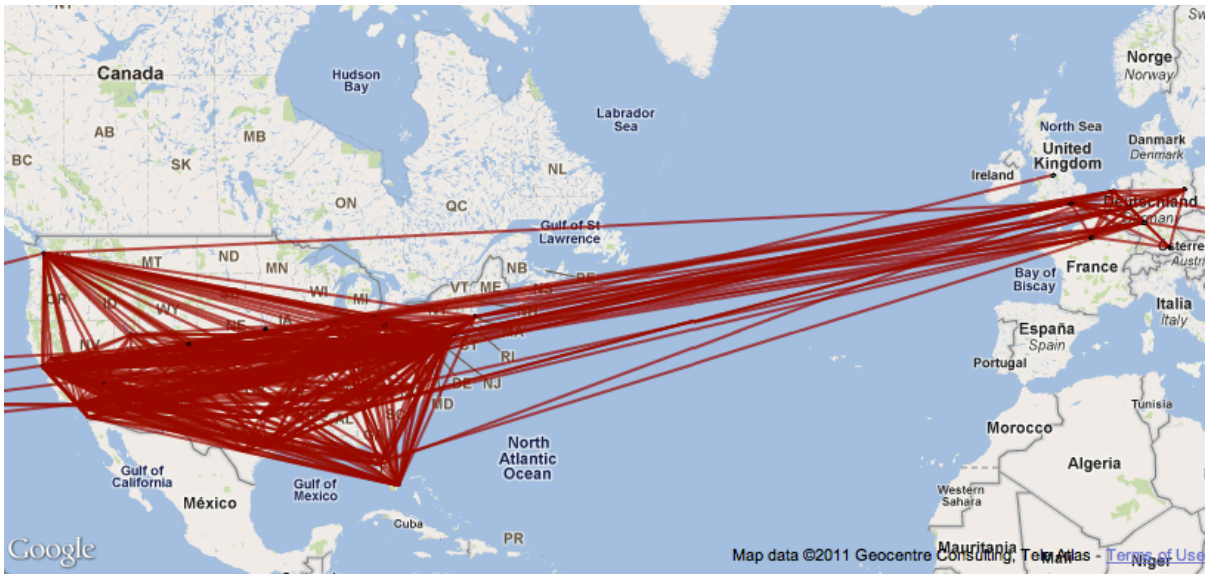

Figure C.7: Level-3 logical topology

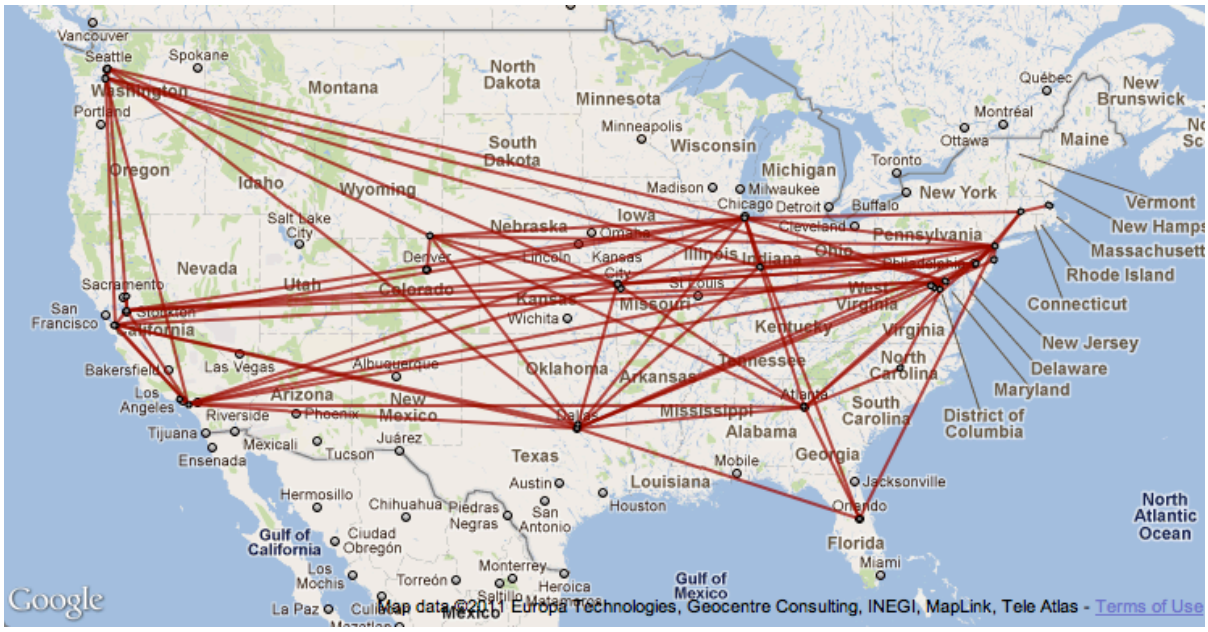

Figure C.8: Sprint logical topology

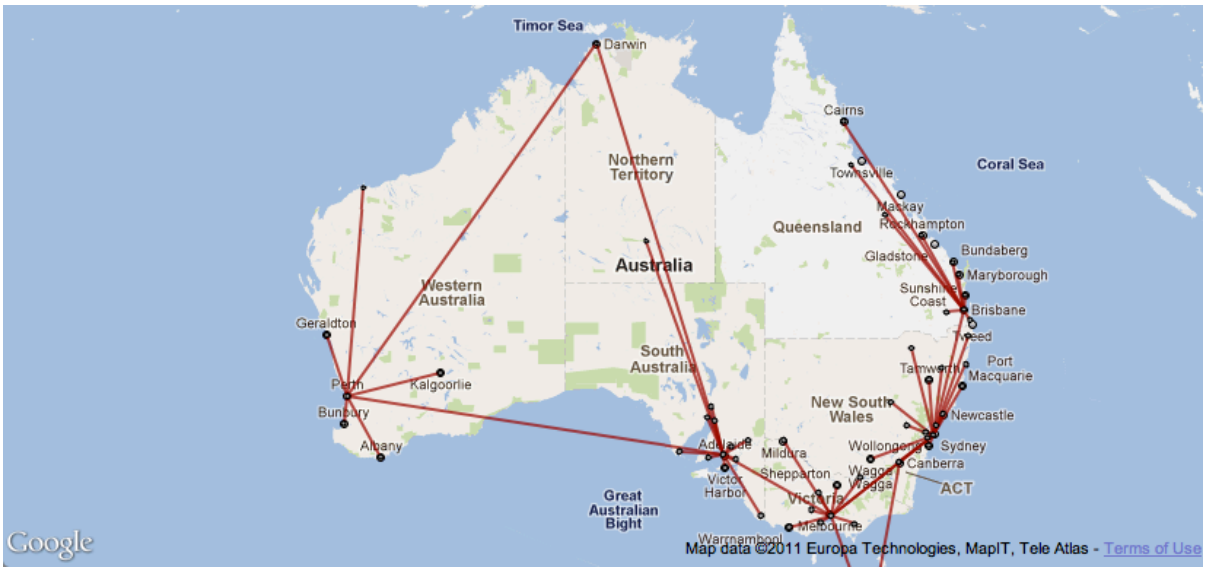

Figure C.9: Telstra logical topology

Figure C.10: Tiscali logical topology

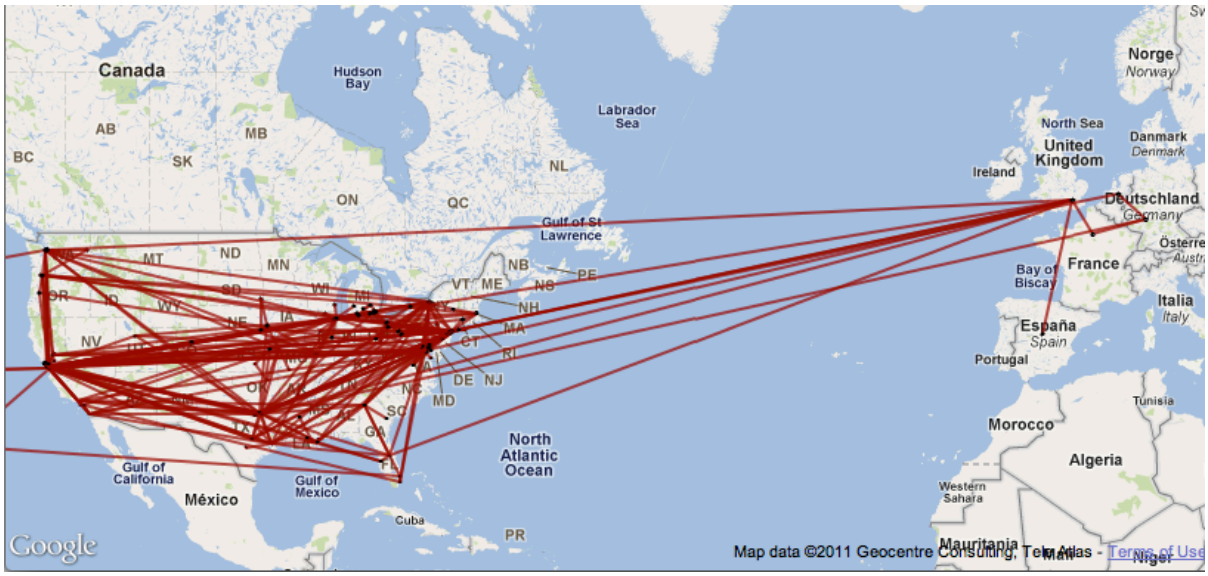

Figure C.11: Verio logical topology

Figure C.12: VSNL logical topology

Appendix D

Dynamic Graph Properties

This appendix contains a full set of of plots characterizing various graph properties recalculated after node and link failures occur with varying probabilities.

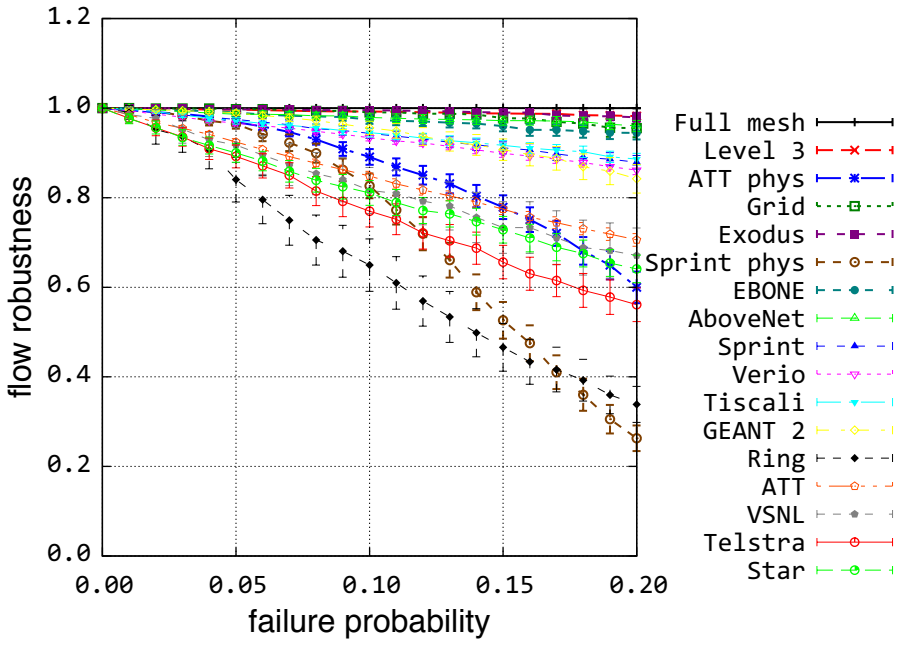

Figure D.1: Connectivity vs. link failure probability for all topologies

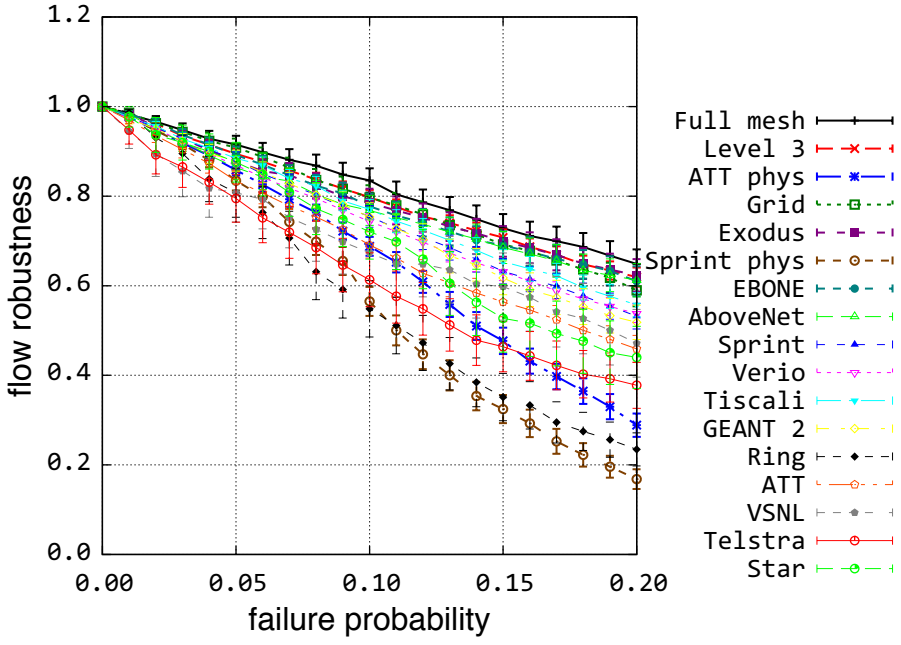

Figure D.2: Connectivity vs. node failure probability for all topologies

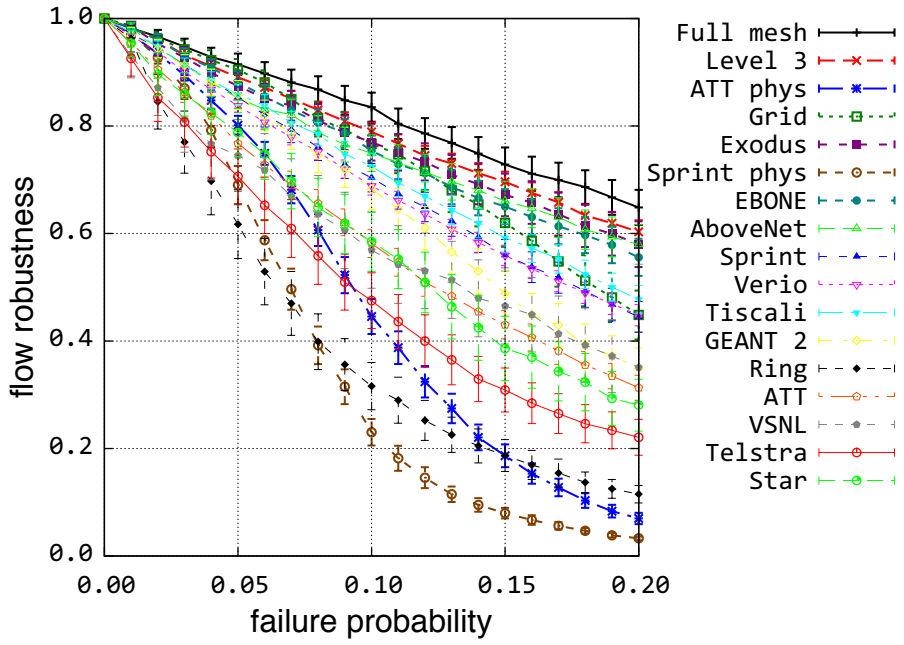

Figure D.3: Connectivity vs. node & link failure probability for all topologies

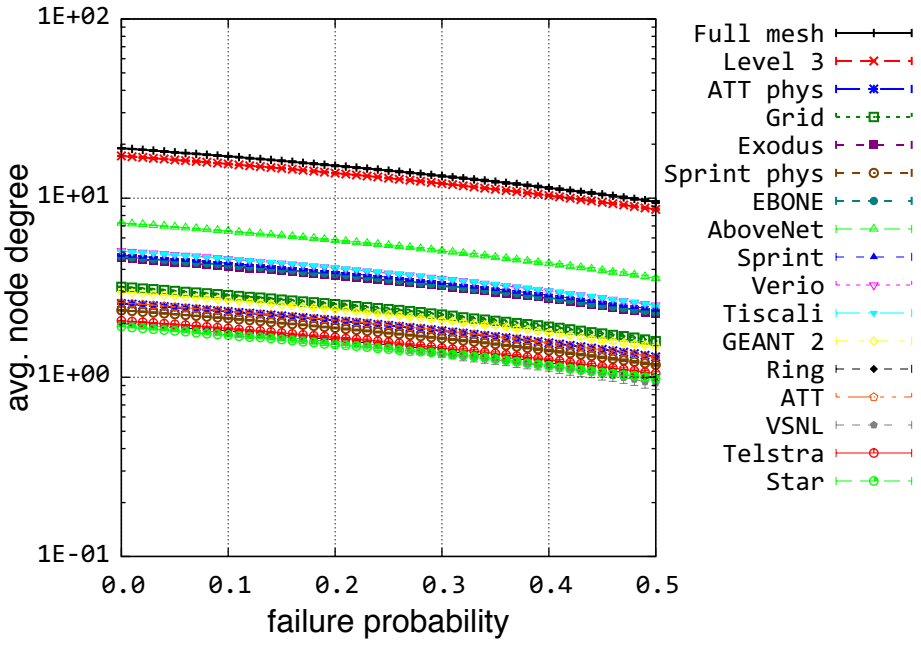

Figure D.4: Average node degree vs. link failure probability for all topologies

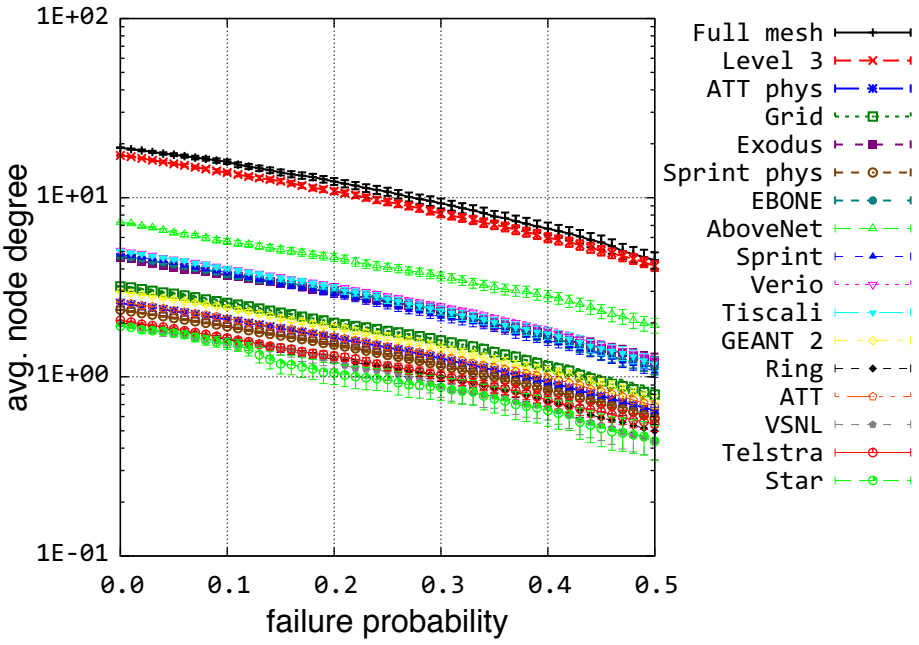

Figure D.5: Average node degree vs. node failure probability for all topologies

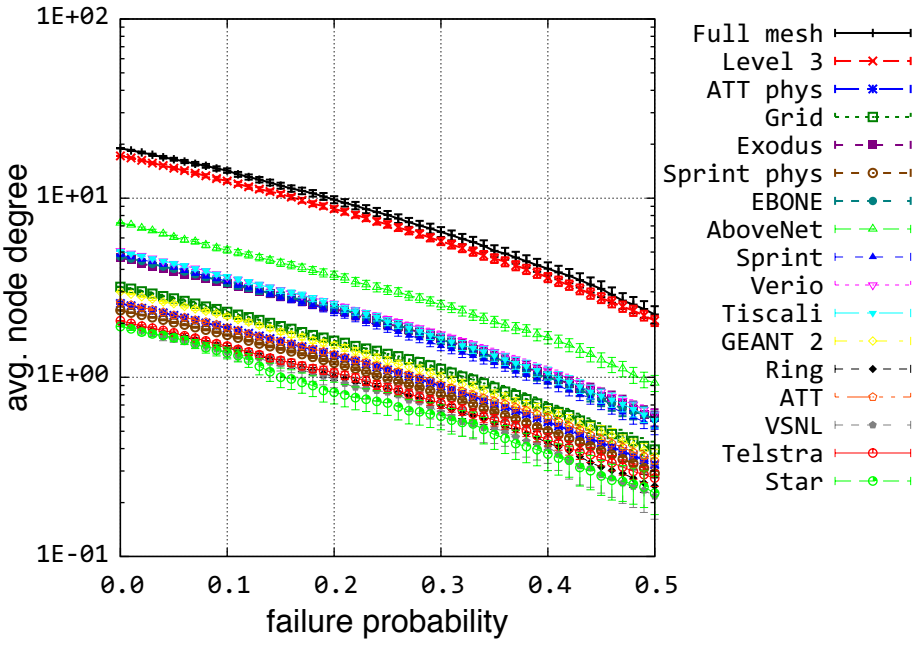

Figure D.6: Average node degree vs. node & link failure probability for all topologies

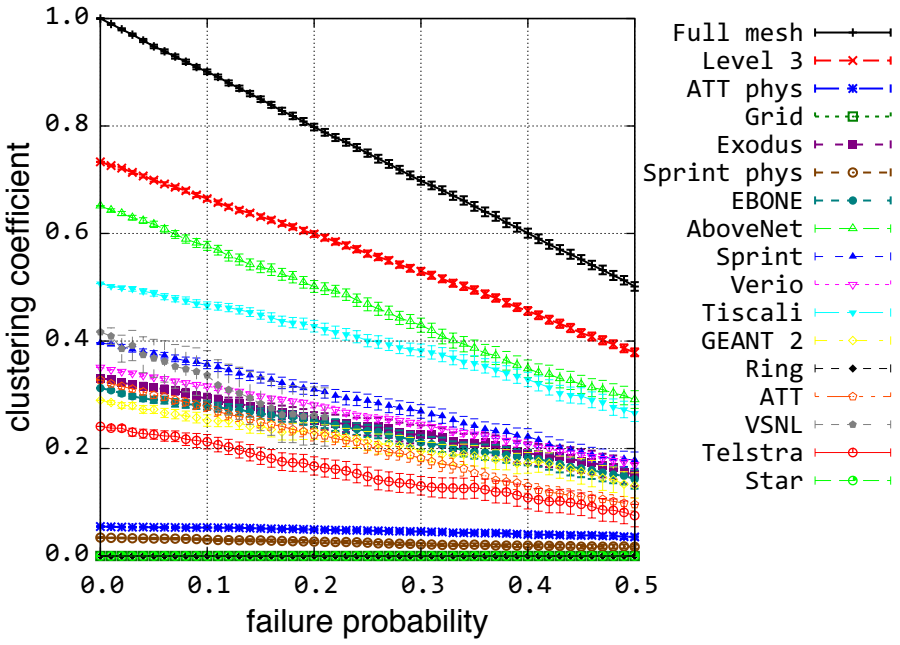

Figure D.7: Clustering coefficient vs. link failure probability for all topologies

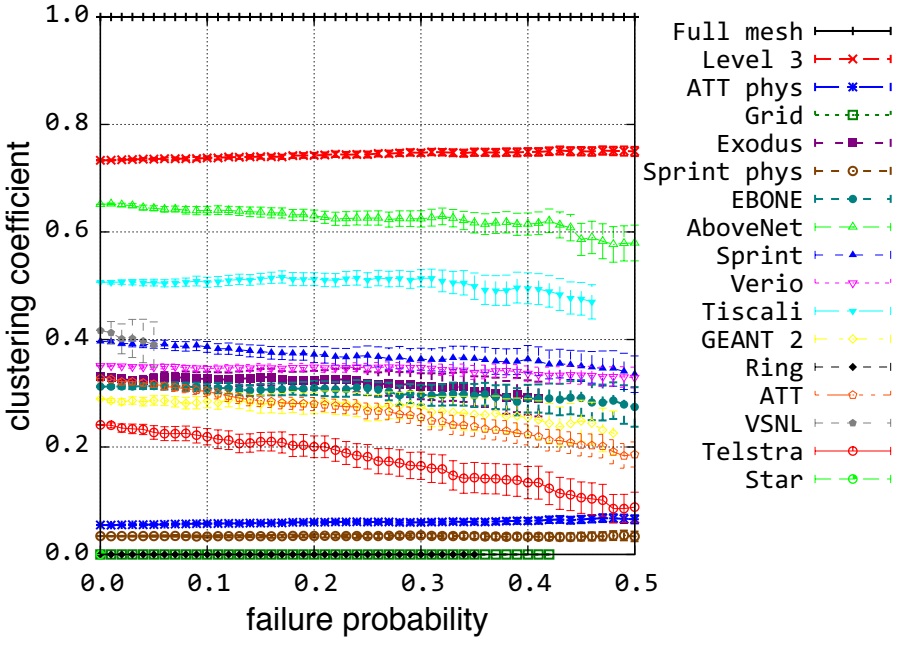

Figure D.8: Clustering coefficient vs. node failure probability for all topologies

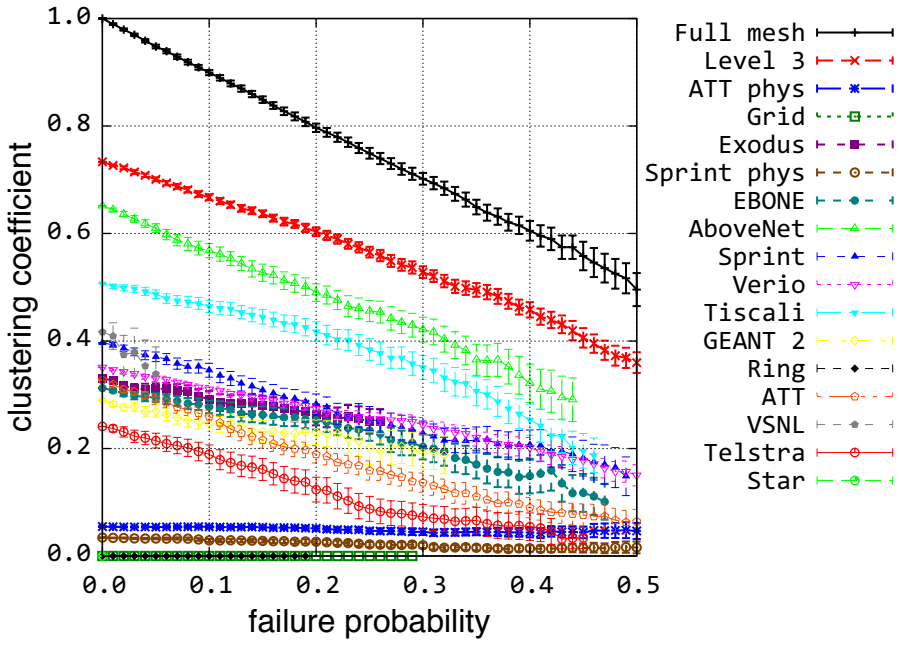

Figure D.9: Clustering coefficient vs. node & link failure probability for all topologies

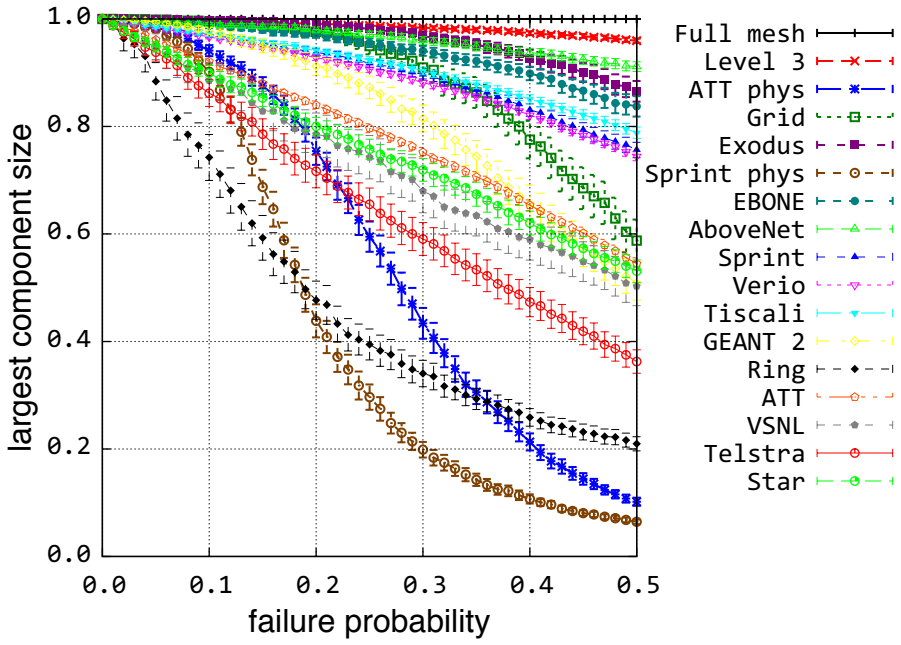

Figure D.10: Largest component size vs. link failure probability for all topologies

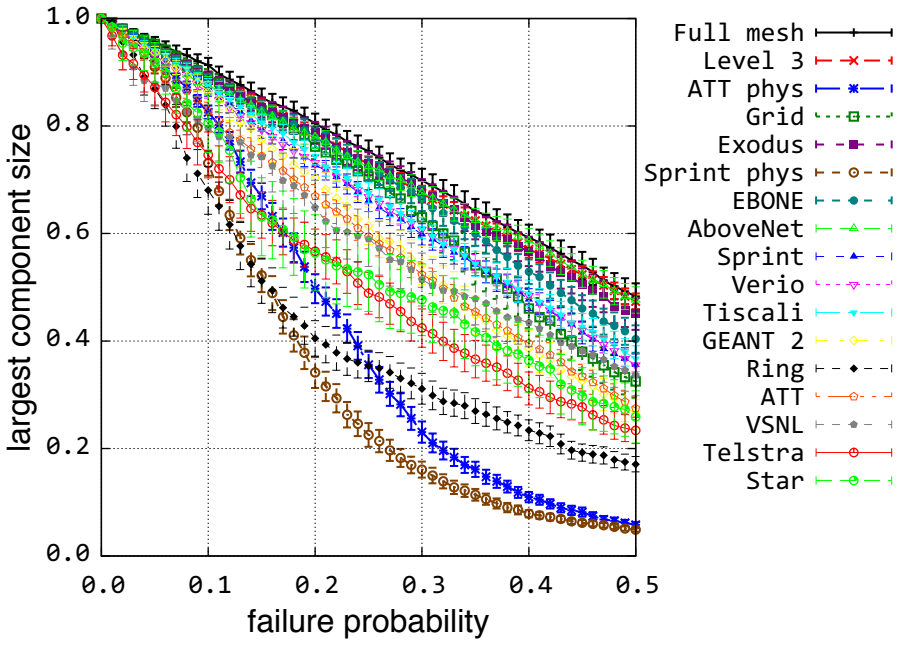

Figure D.11: Largest component size vs. node failure probability for all topologies

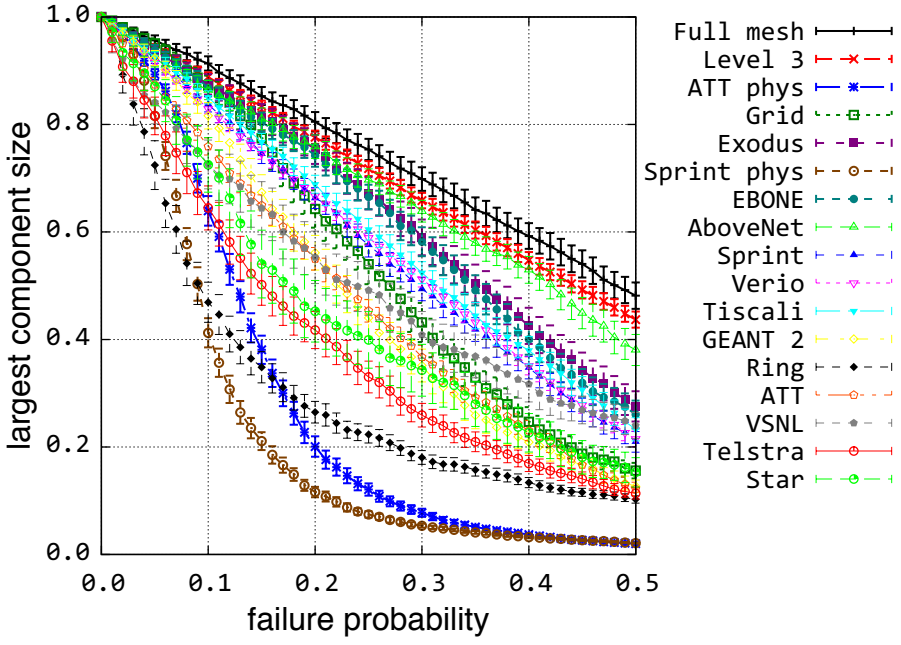

Figure D.12: Largest component size vs. node & link failure probability for all topologies

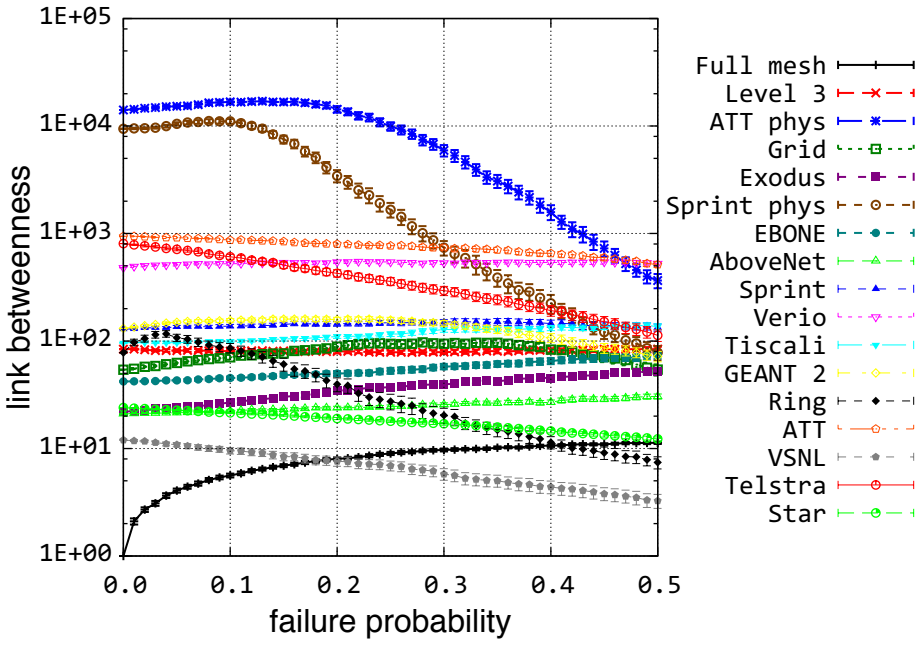

Figure D.13: Link betweenness vs. link failure probability for all topologies

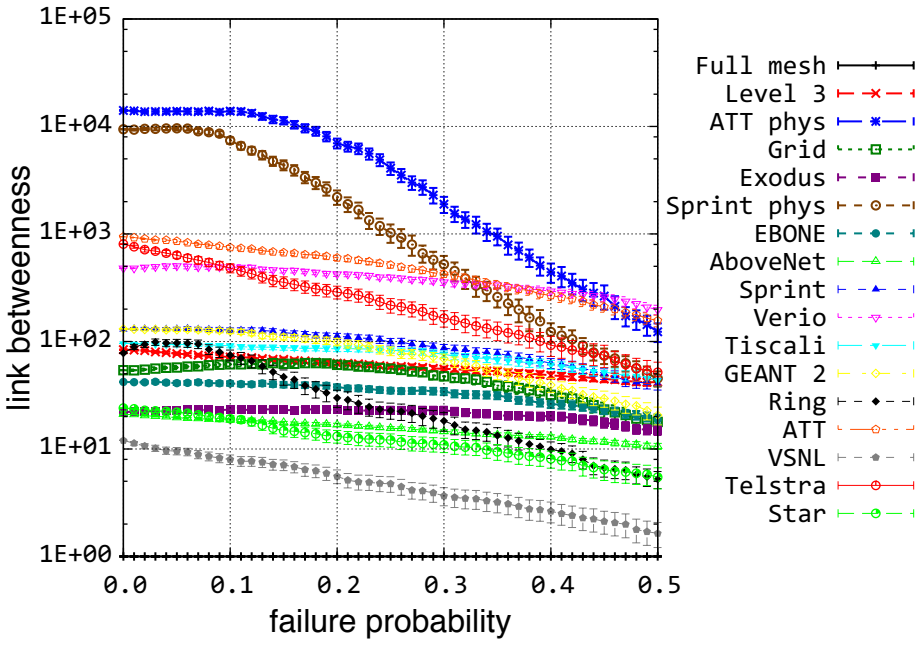

Figure D.14: Link betweenness vs. node failure probability for all topologies

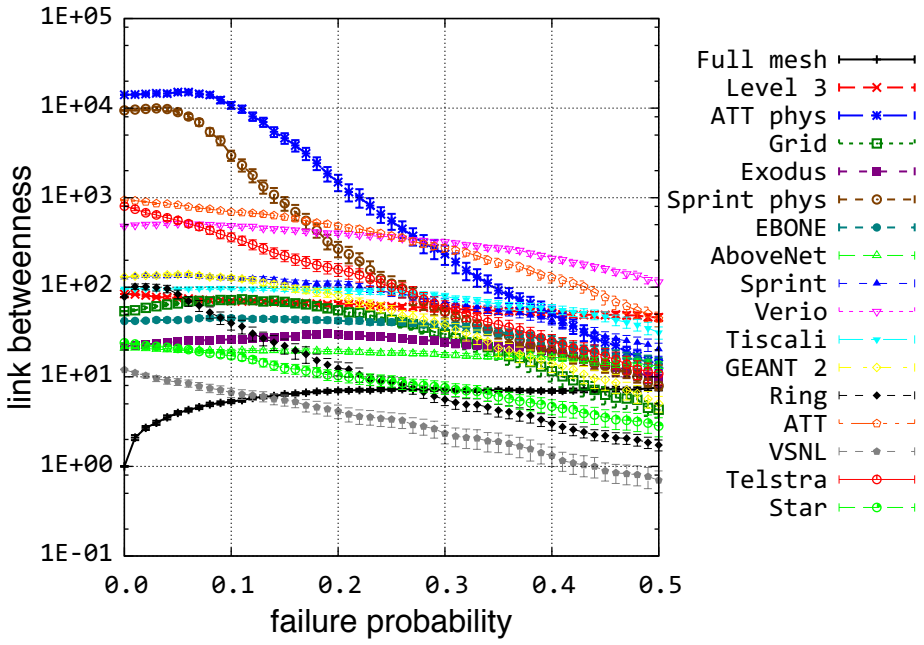

Figure D.15: Link betweenness vs. node & link failure probability for all topologies

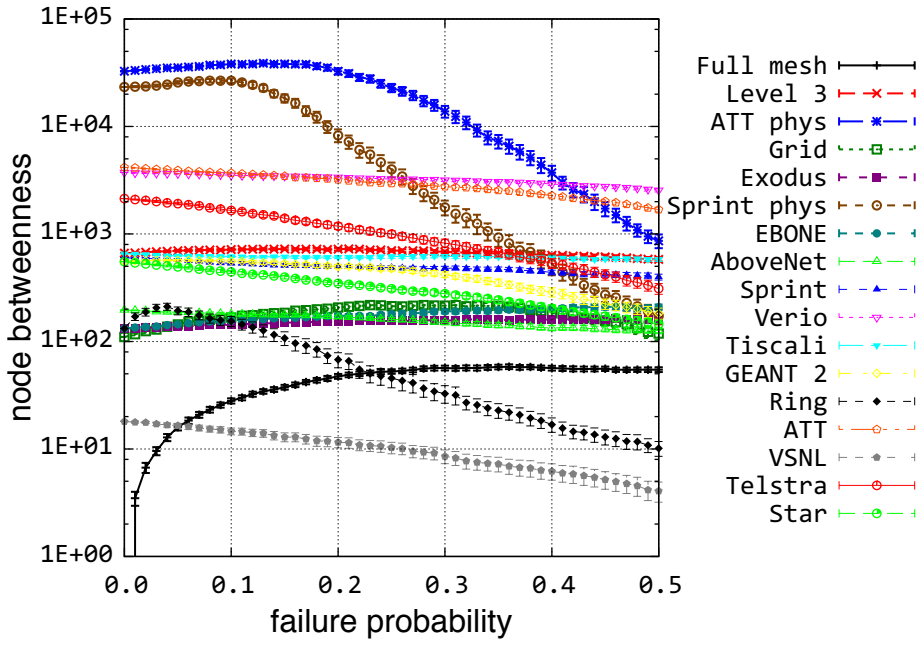

Figure D.16: Node betweenness vs. link failure probability for all topologies

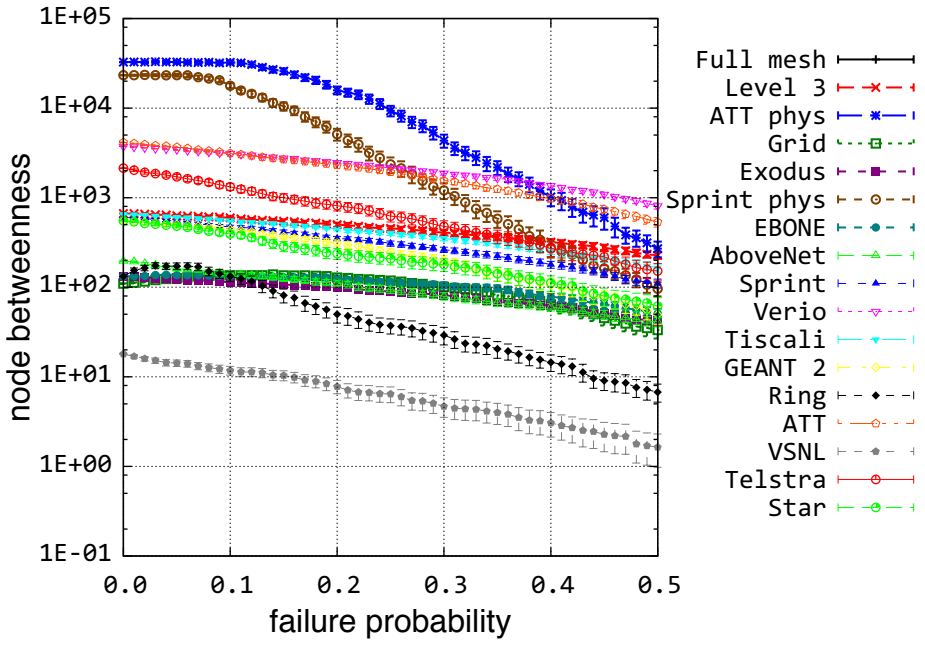

Figure D.17: Node betweenness vs. node failure probability for all topologies

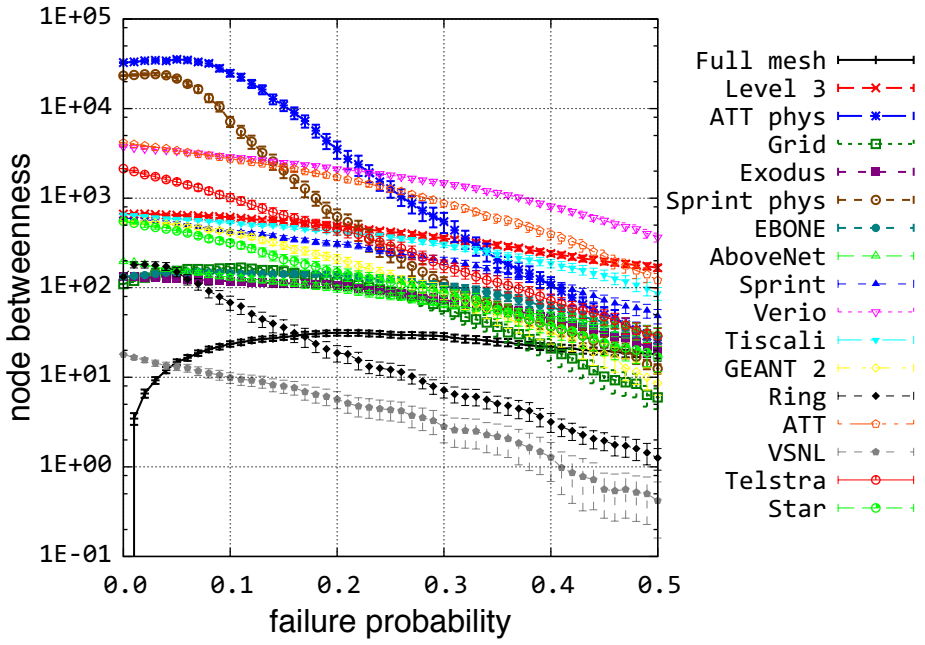

Figure D.18: Node betweenness vs. node & link failure probability for all topologies

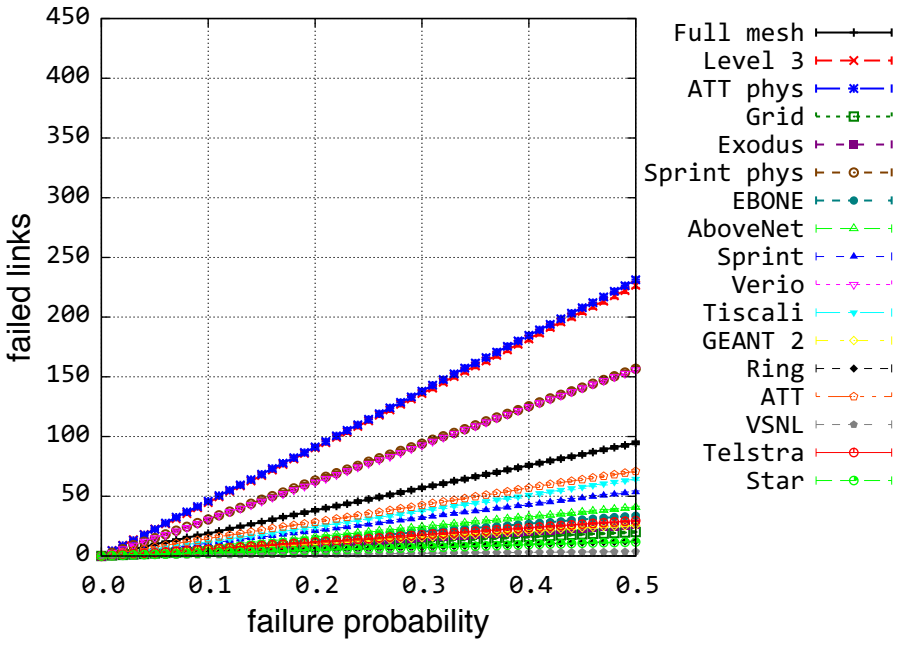

Figure D.19: Number of failed links vs. link failure probability for all topologies

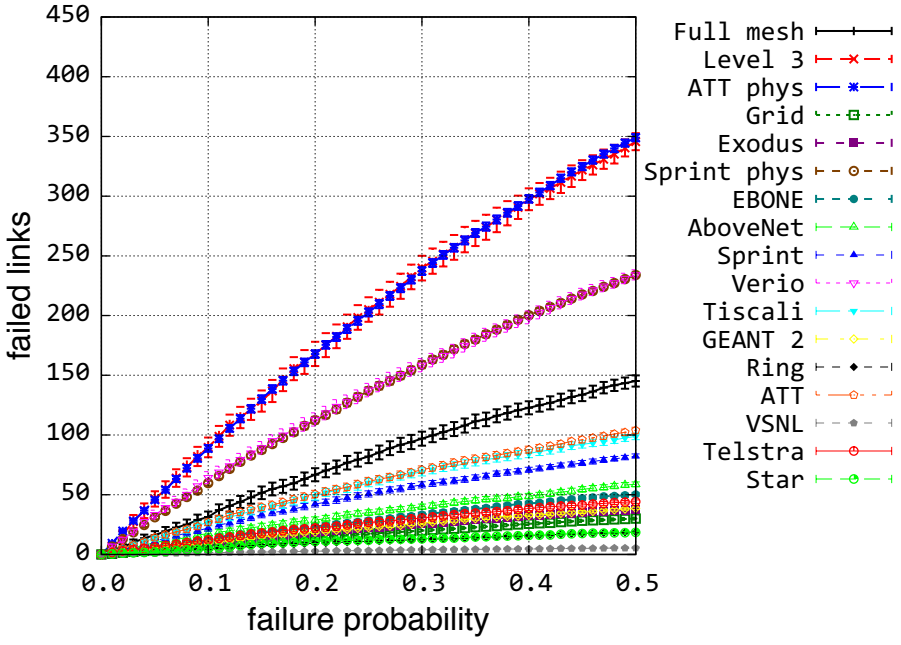

Figure D.20: Number of failed links vs. node failure probability for all topologies

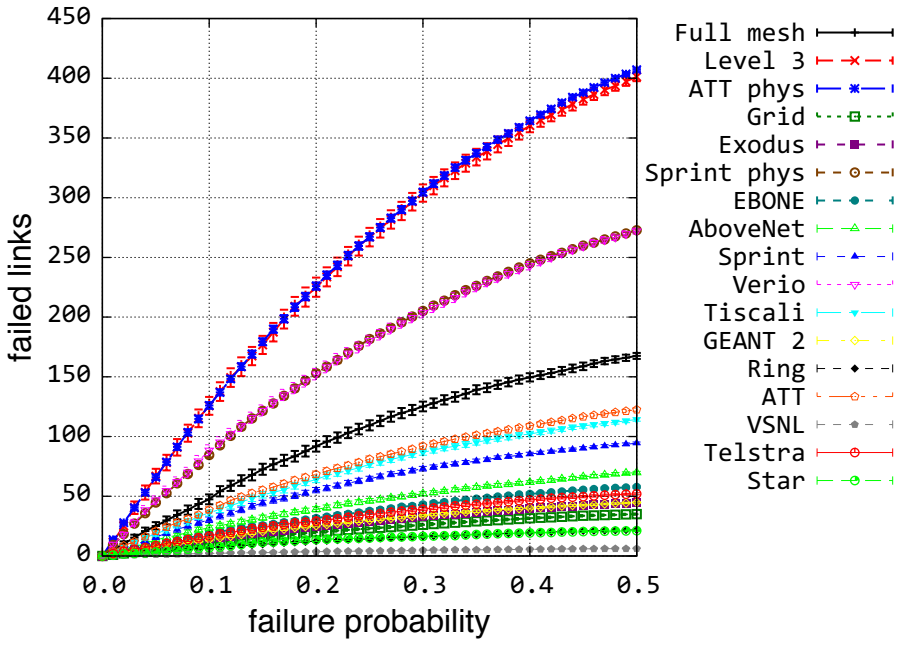

Figure D.21: Number of failed links vs. node & link failure probability for all topologies

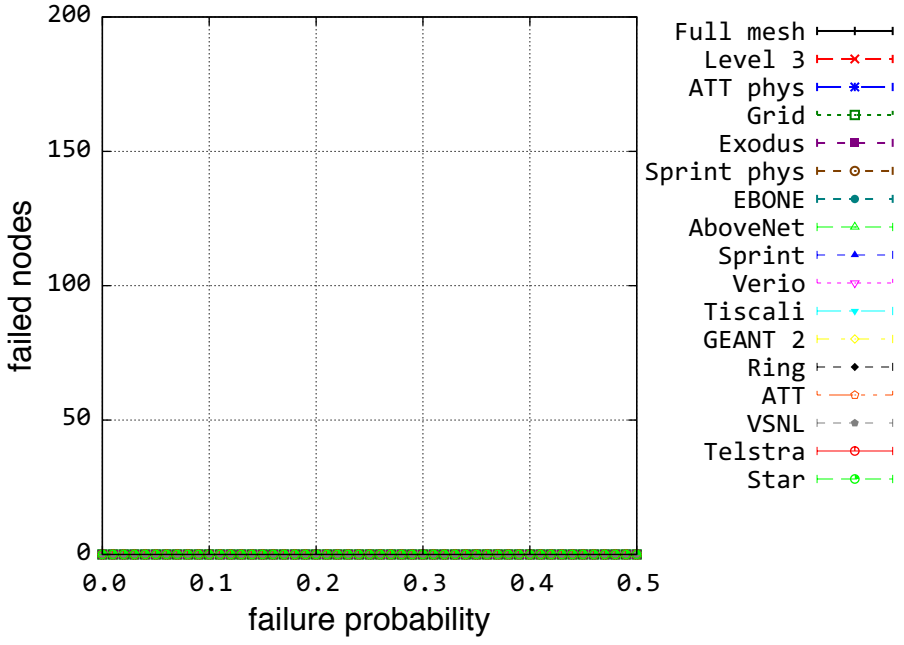

Figure D.22: Number of failed nodes vs. link failure probability for all topologies

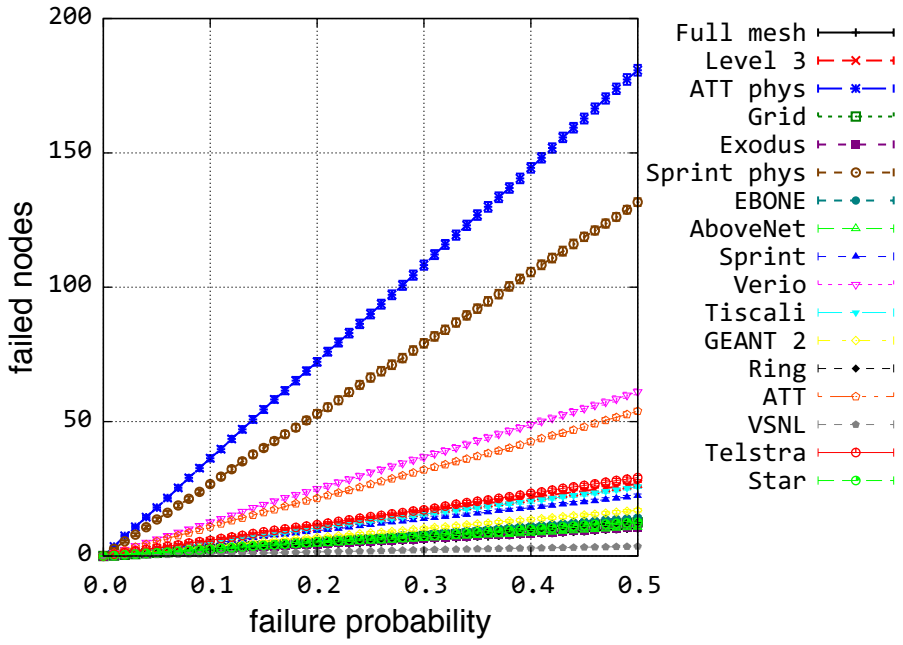

Figure D.23: Number of failed nodes vs. node failure probability for all topologies

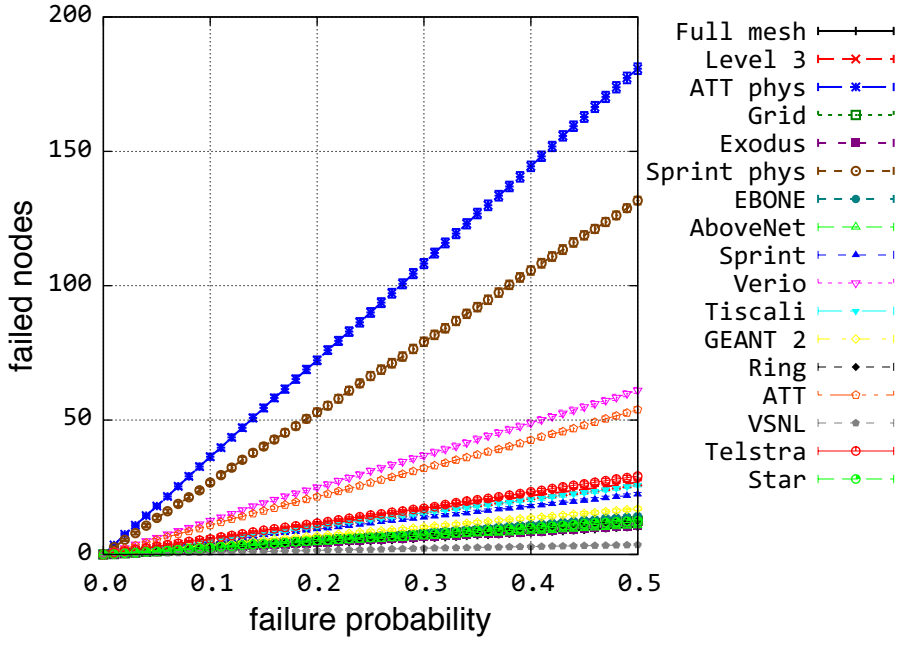

Figure D.24: Number of failed nodes vs. node & link failure probability for all topologies

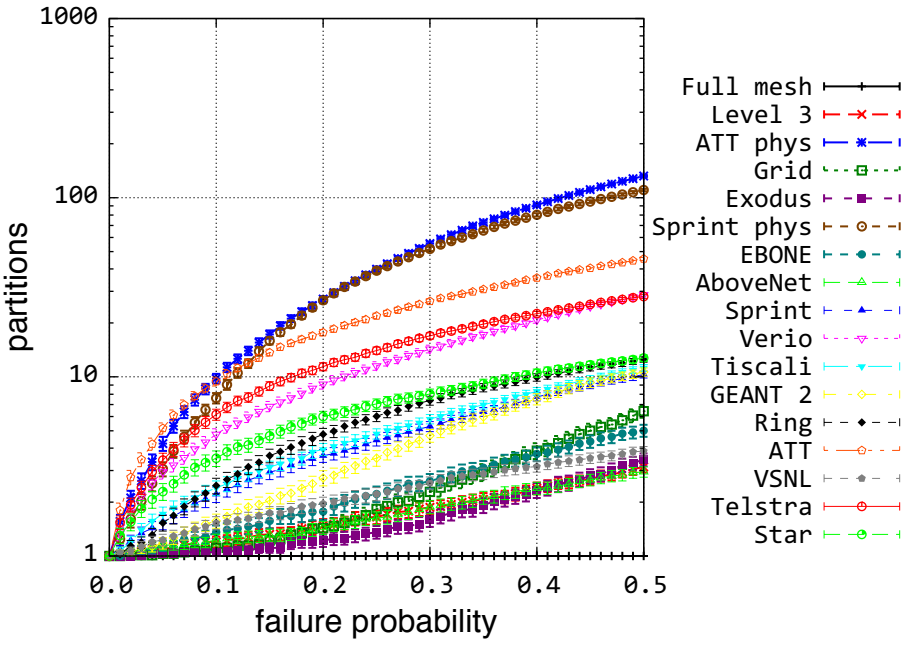

Figure D.25: Number of partitions vs. link failure probability for all topologies

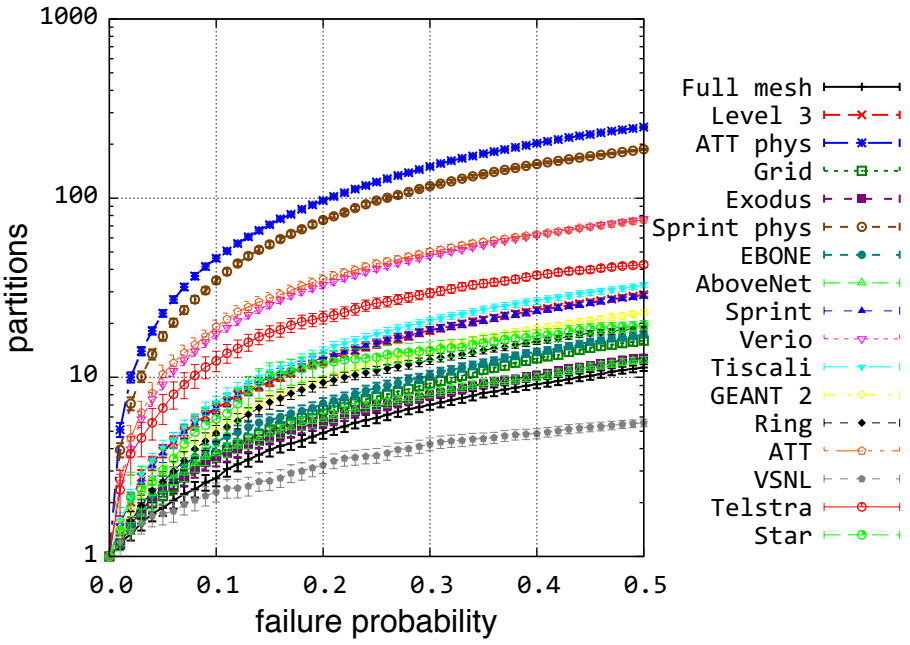

Figure D.26: Number of partitions vs. node failure probability for all topologies

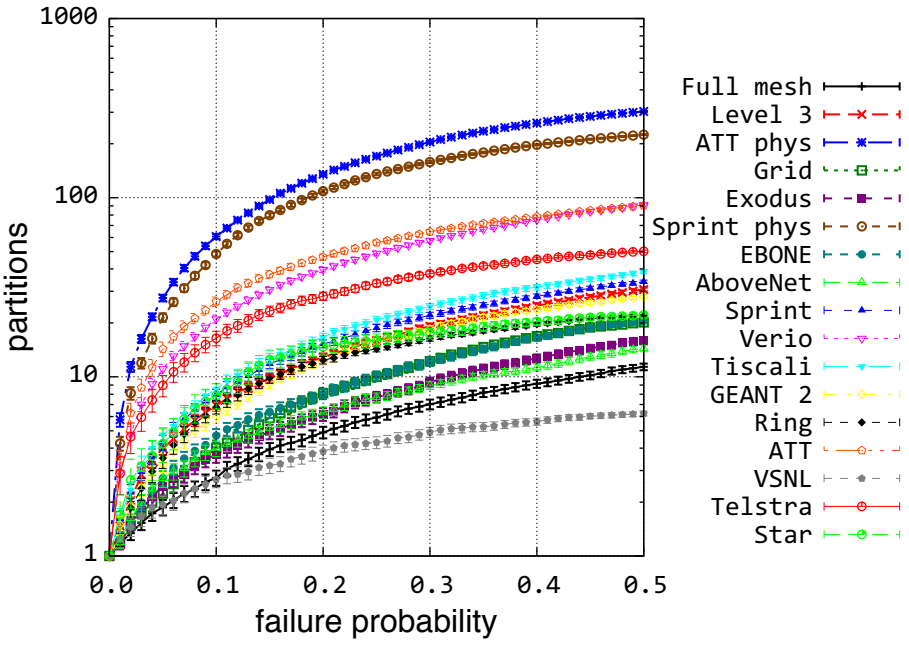

Figure D.27: Number of partitions vs. node & link failure probability for all topologies

Page left intentionally blank.

Appendix E

Flow-Robustness Plots

This appendix contains a full set of flow-robustness plots for the topologies used in the analysis of the *path diversification* metrics.

E.1 Physical Topologies

E.1.1 AT&T physical

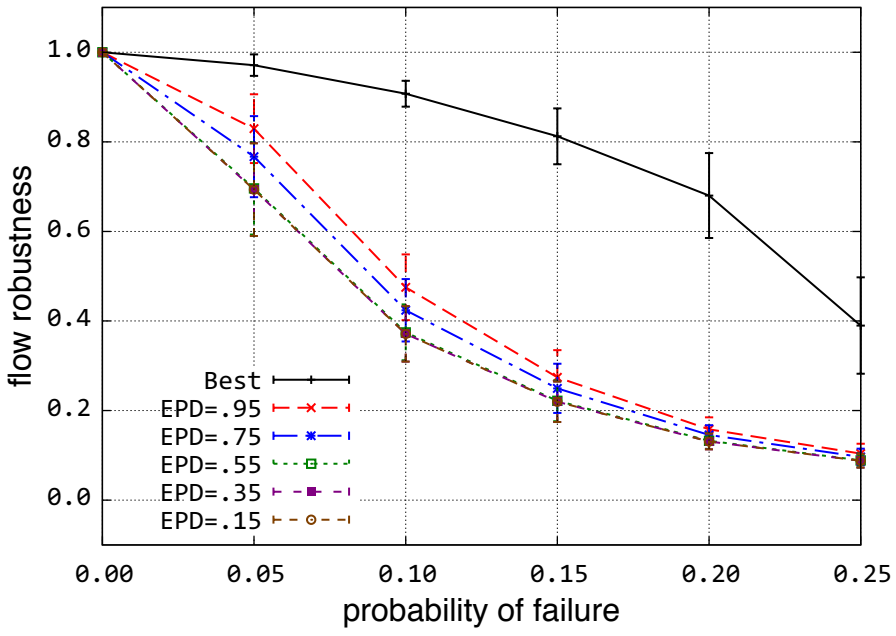

Figure E.1: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the AT&T physical topology

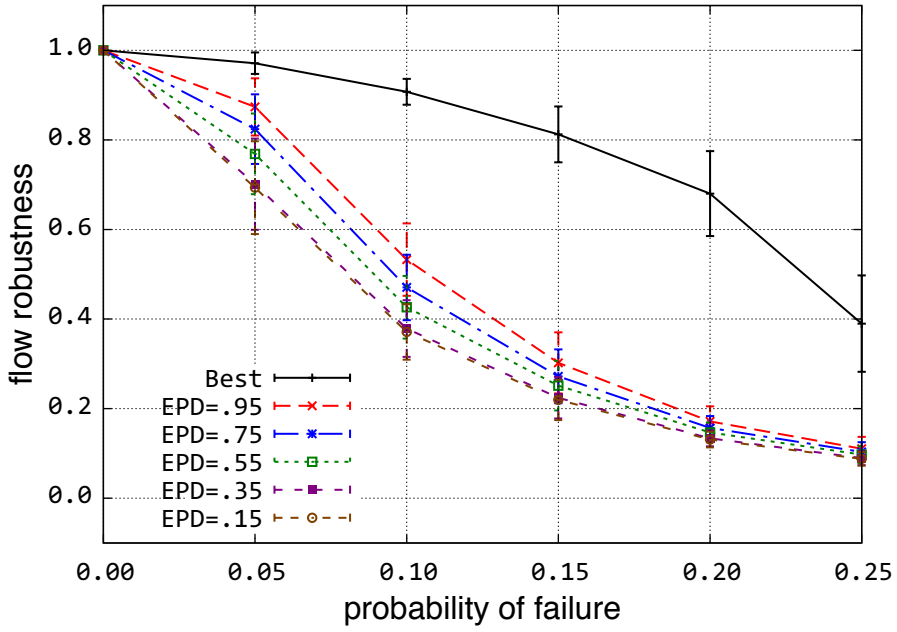

Figure E.2: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the AT&T physical topology

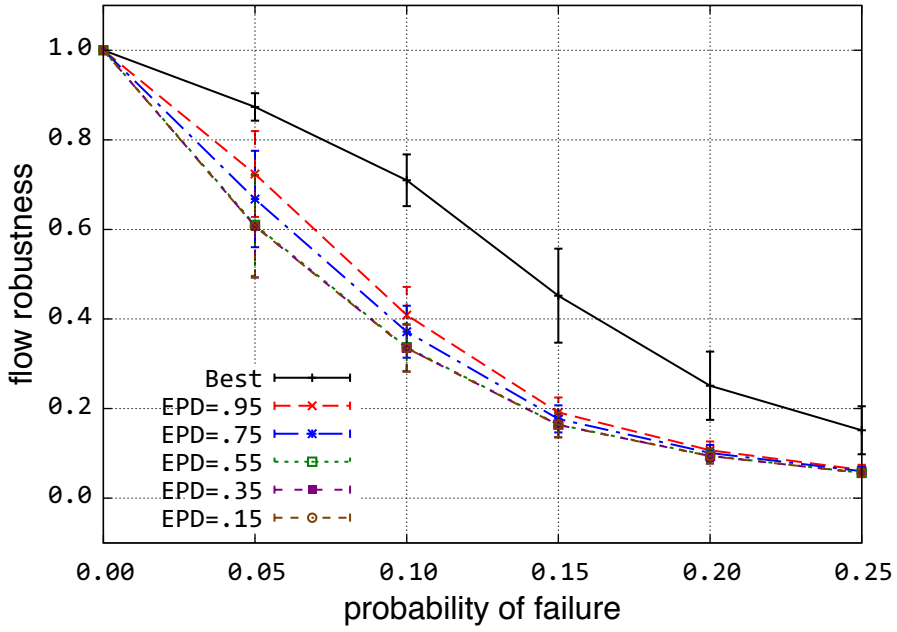

Figure E.3: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the AT&T physical topology

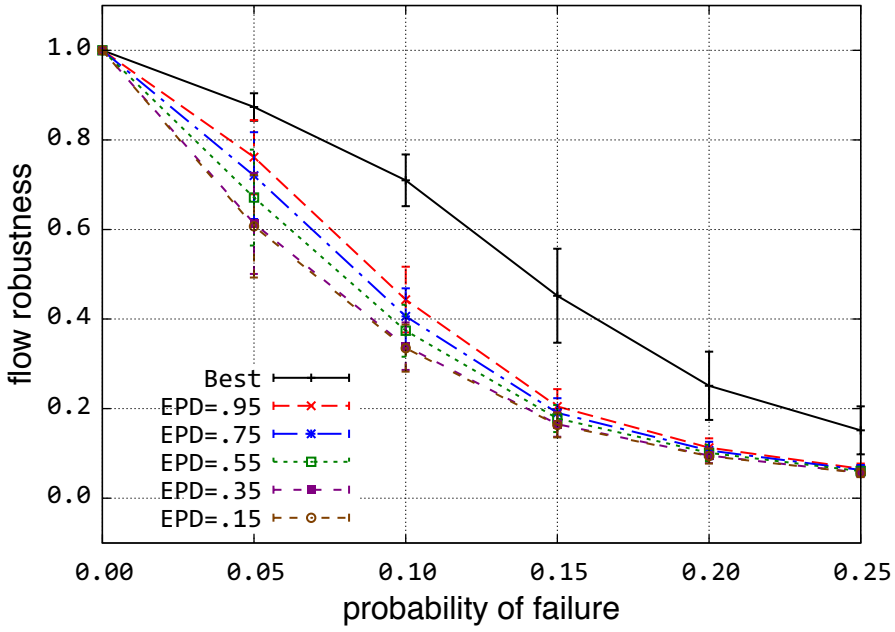

Figure E.4: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the AT&T physical topology

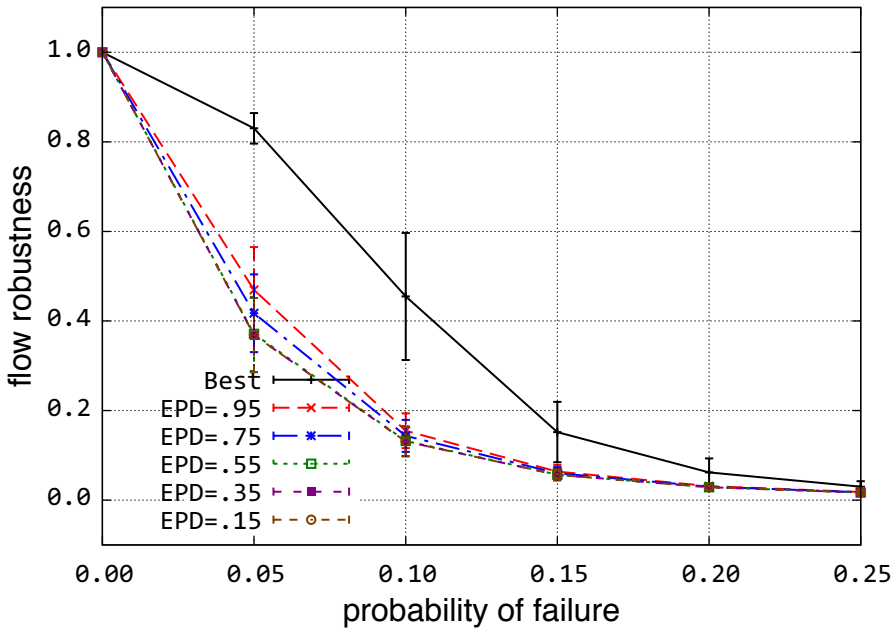

Figure E.5: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the AT&T physical topology

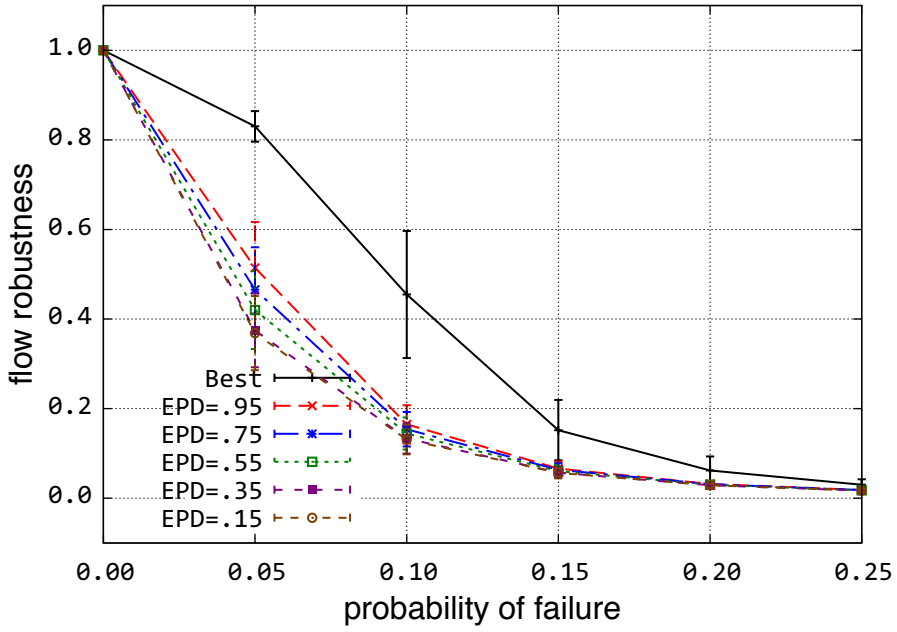

Figure E.6: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the AT&T physical topology

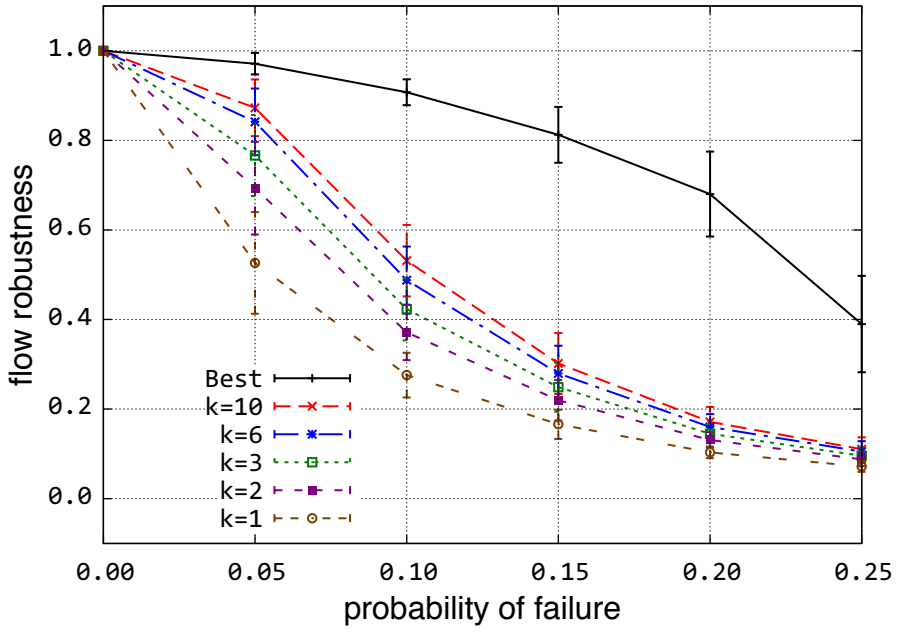

Figure E.7: Flow robustness vs. link failure probability for the AT&T physical topology

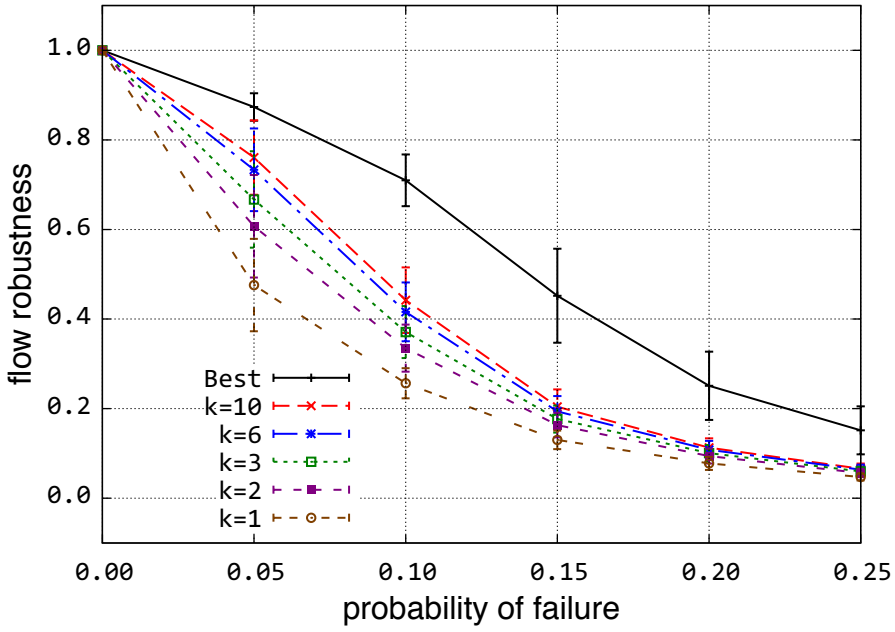

Figure E.8: Flow robustness vs. node failure probability for the AT&T physical topology

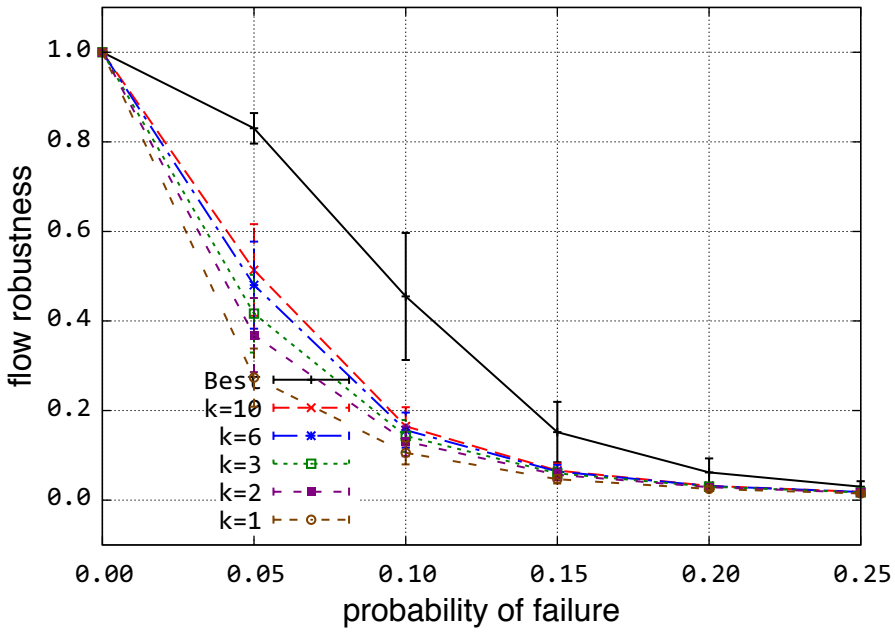

Figure E.9: Flow robustness vs. node & link failure probability for the AT&T physical topology

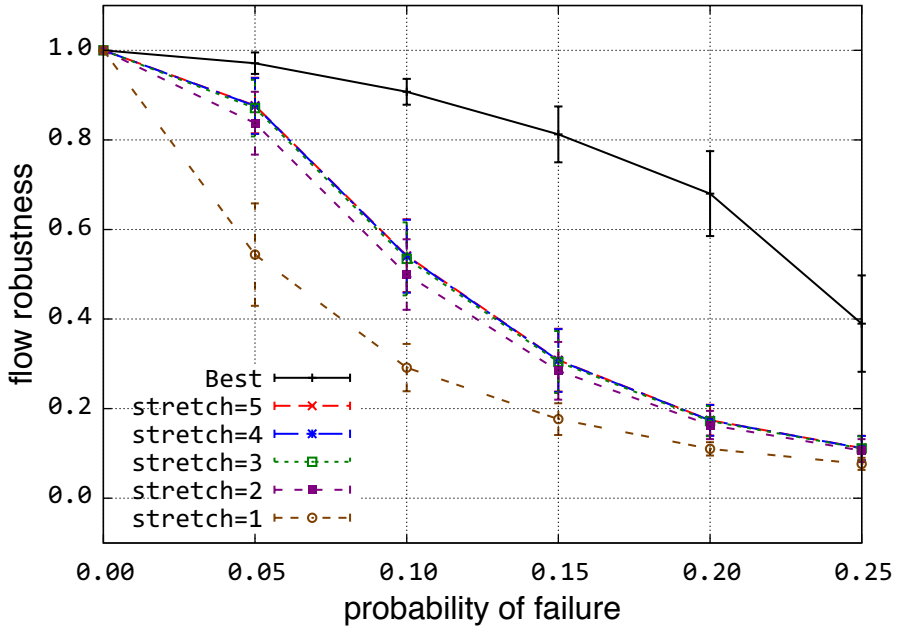

Figure E.10: Flow robustness vs. link failure probability for various stretch limits on the AT&T physical topology

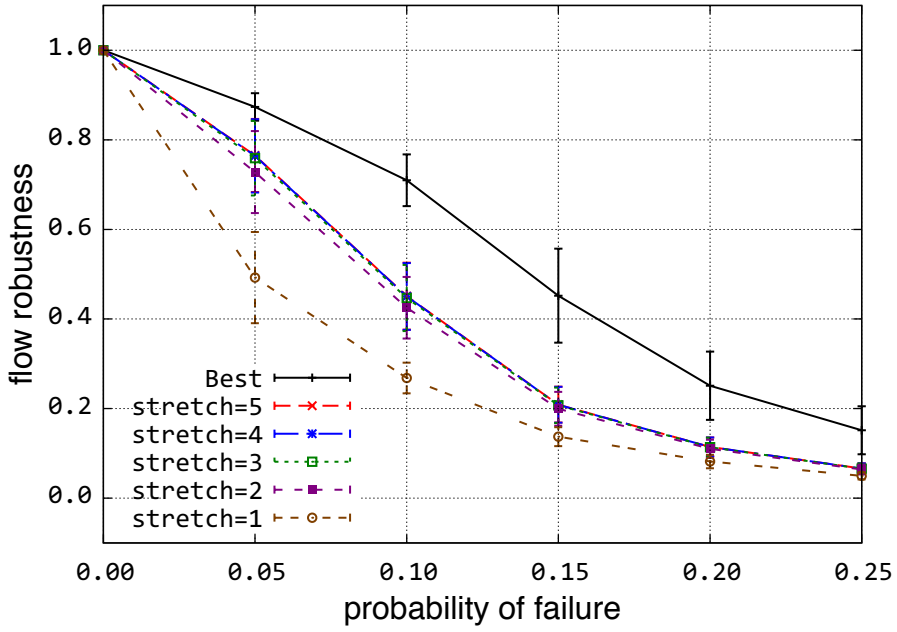

Figure E.11: Flow robustness vs. node failure probability for various stretch limits on the AT&T physical topology

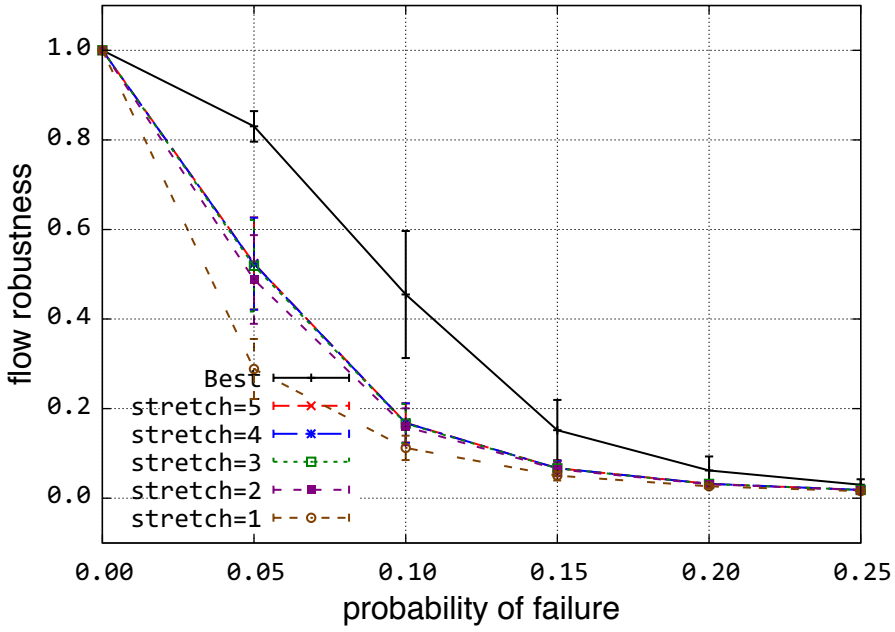

Figure E.12: Flow robustness vs. node & link failure probability for various stretch limits on the AT&T physical topology

E.1.2 GÉANT2 physical

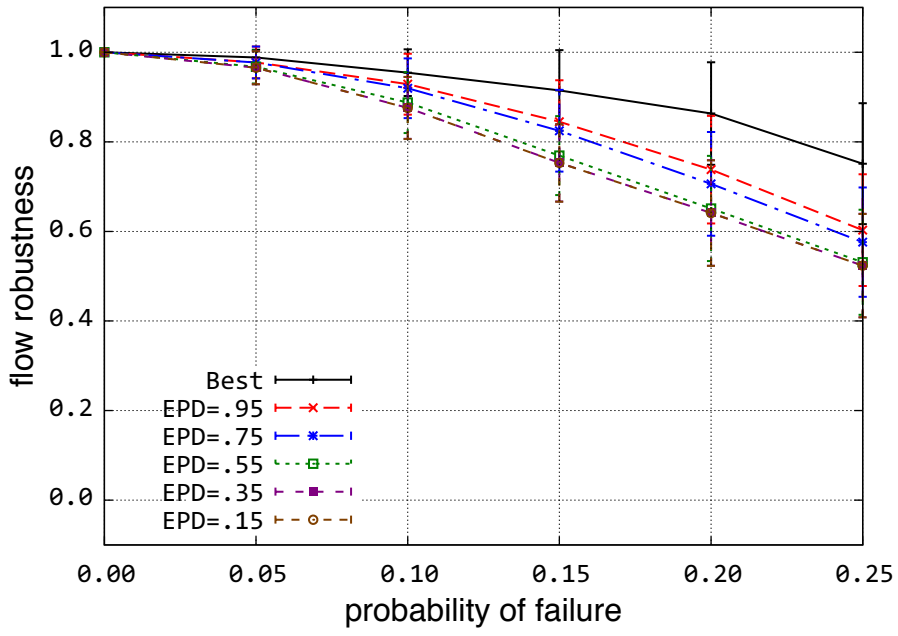

Figure E.13: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the GÉANT2 physical topology

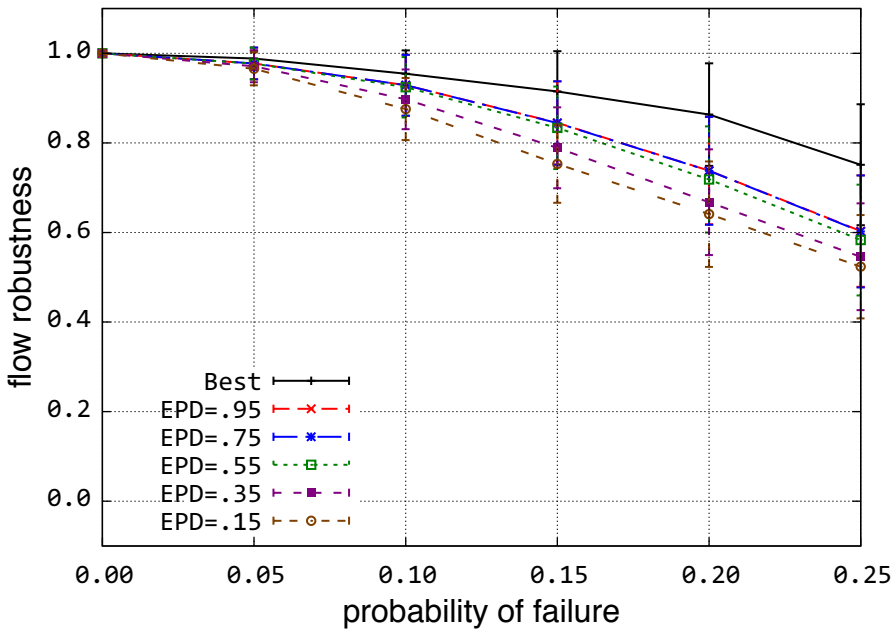

Figure E.14: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the GÉANT2 physical topology

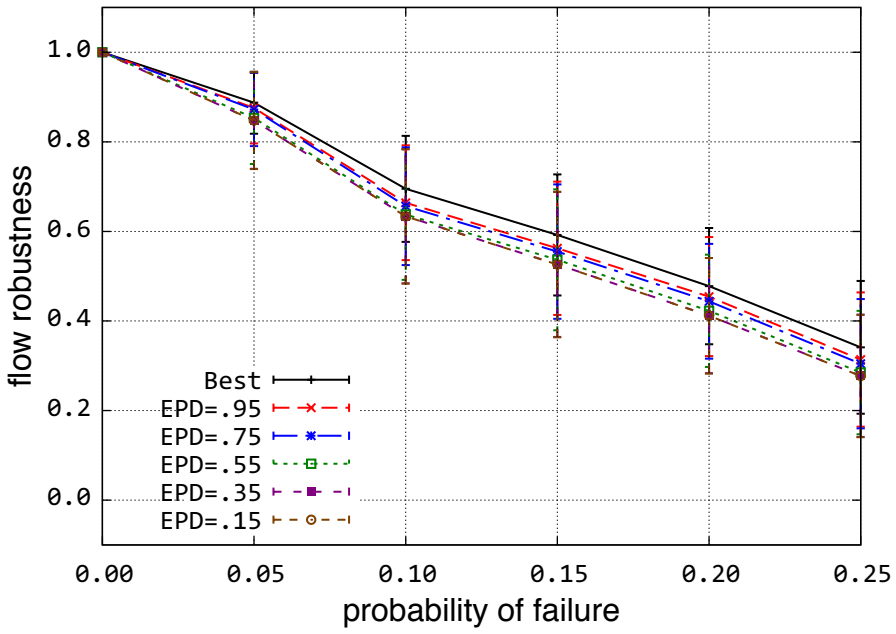

Figure E.15: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the GÉANT2 physical topology

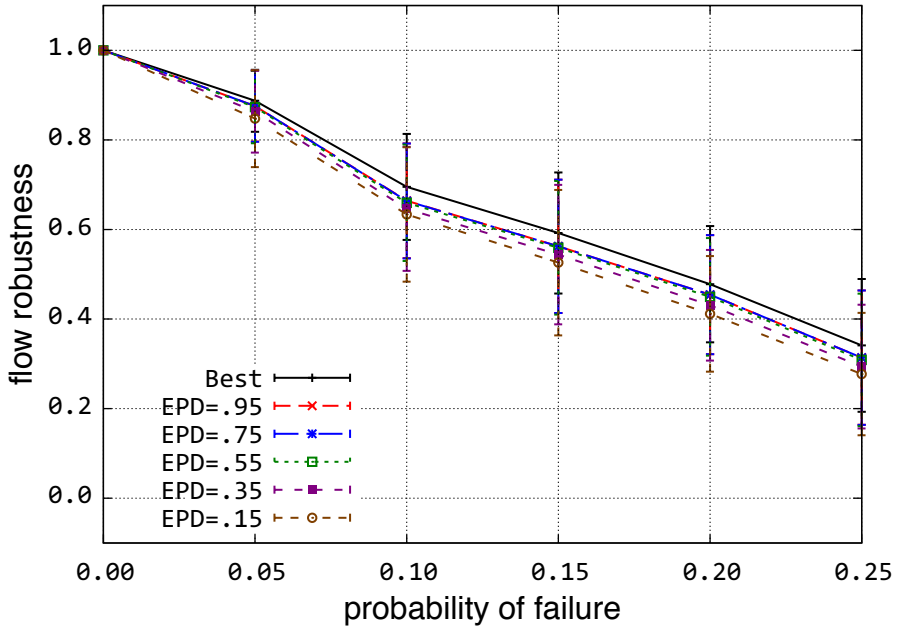

Figure E.16: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the GÉANT2 physical topology

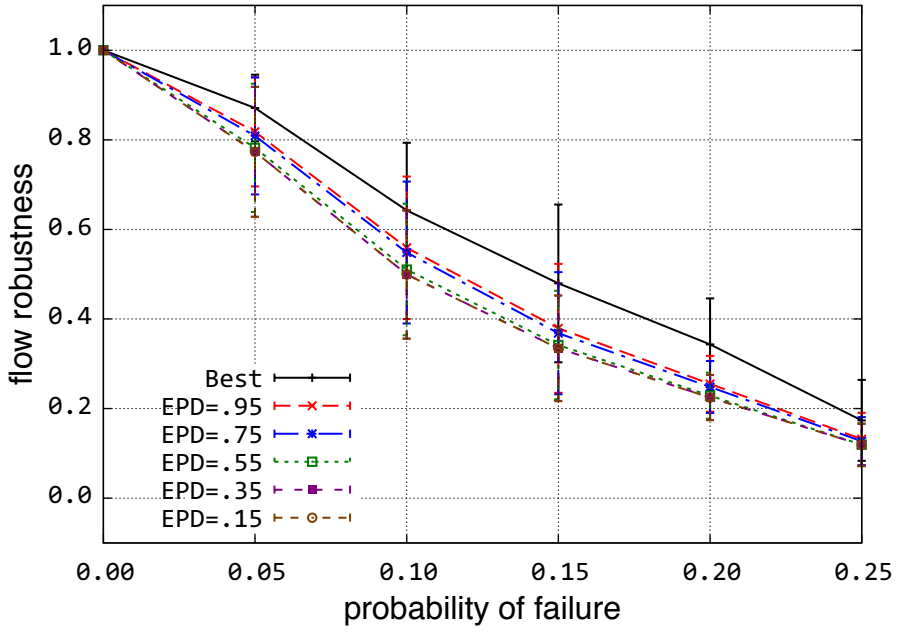

Figure E.17: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the GÉANT2 physical topology

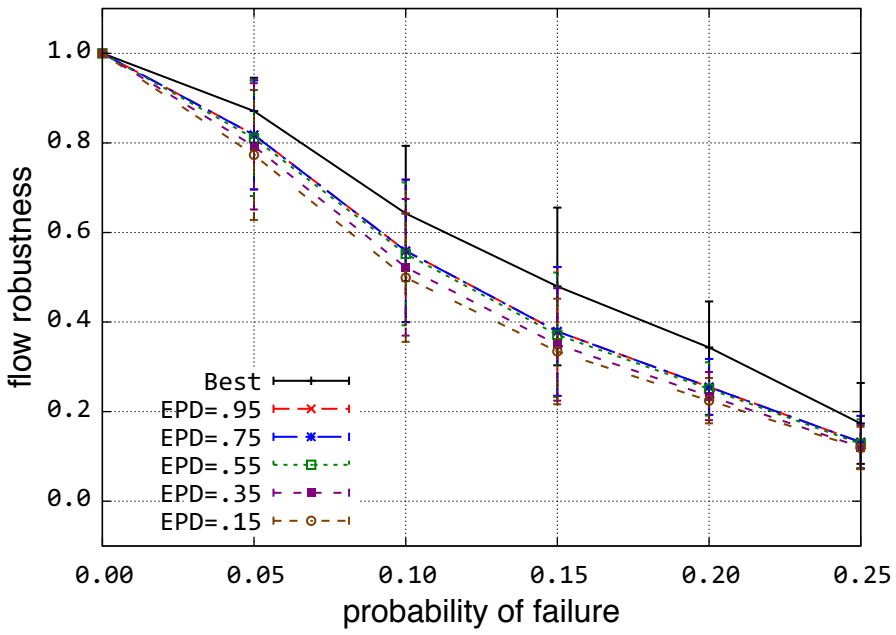

Figure E.18: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the GÉANT2 physical topology

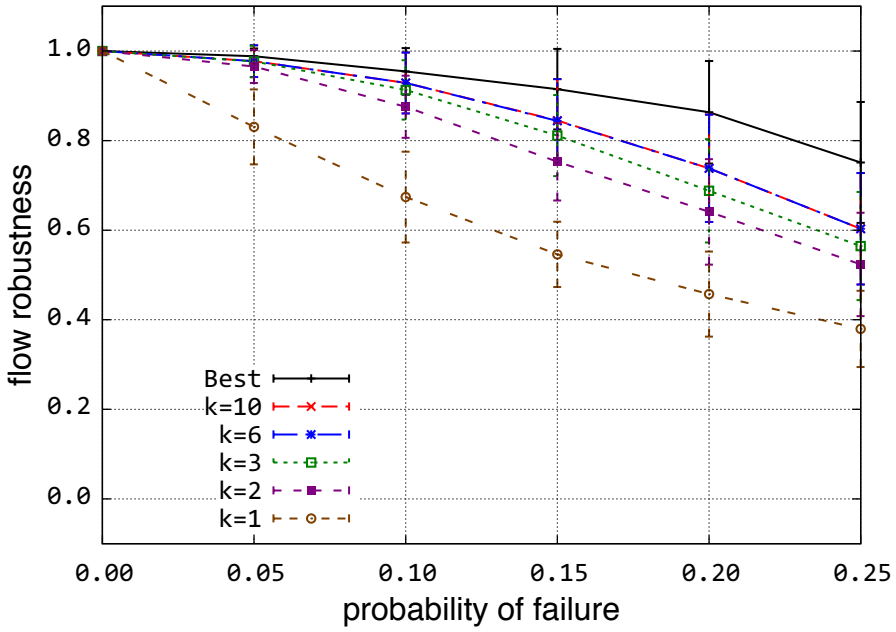

Figure E.19: Flow robustness vs. link failure probability for the GÉANT2 physical topology

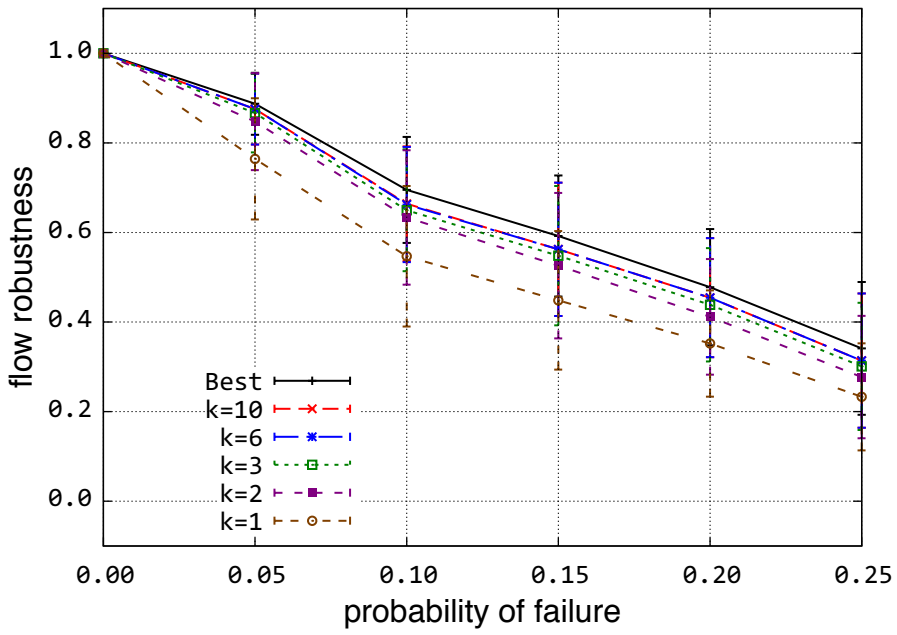

Figure E.20: Flow robustness vs. node failure probability for the GÉANT2 physical topology

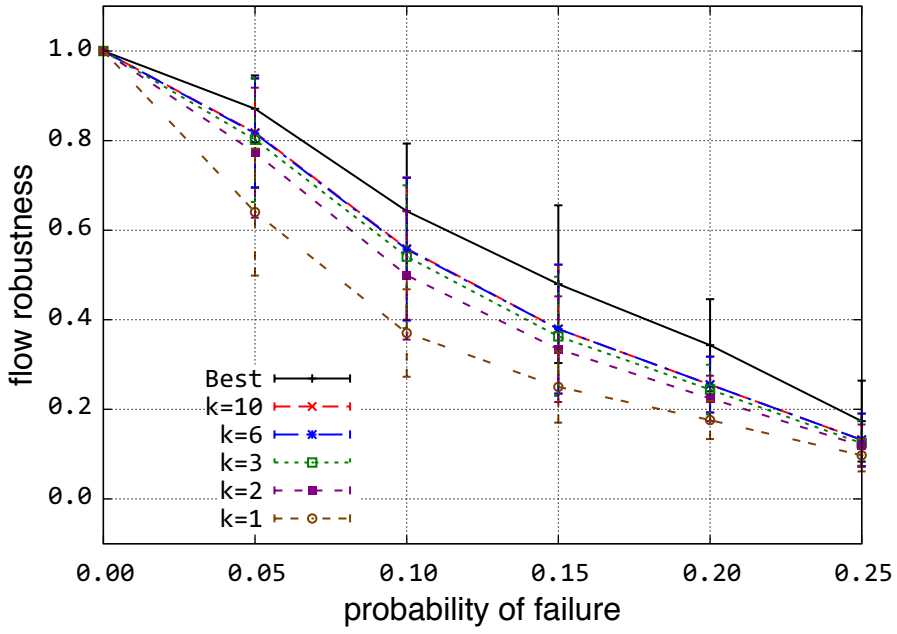

Figure E.21: Flow robustness vs. node & link failure probability for the GÉANT2 physical topology

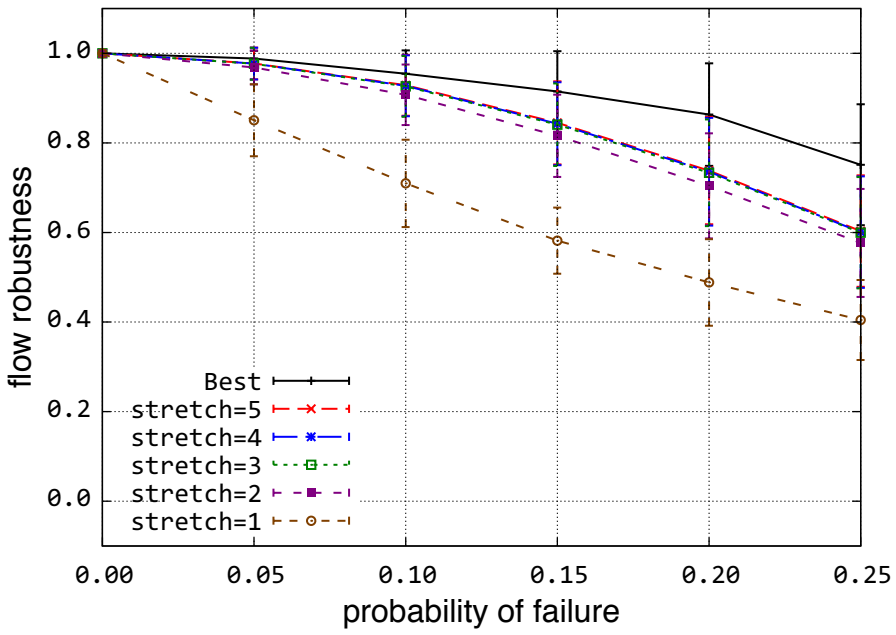

Figure E.22: Flow robustness vs. link failure probability for various stretch limits on the GÉANT2 physical topology

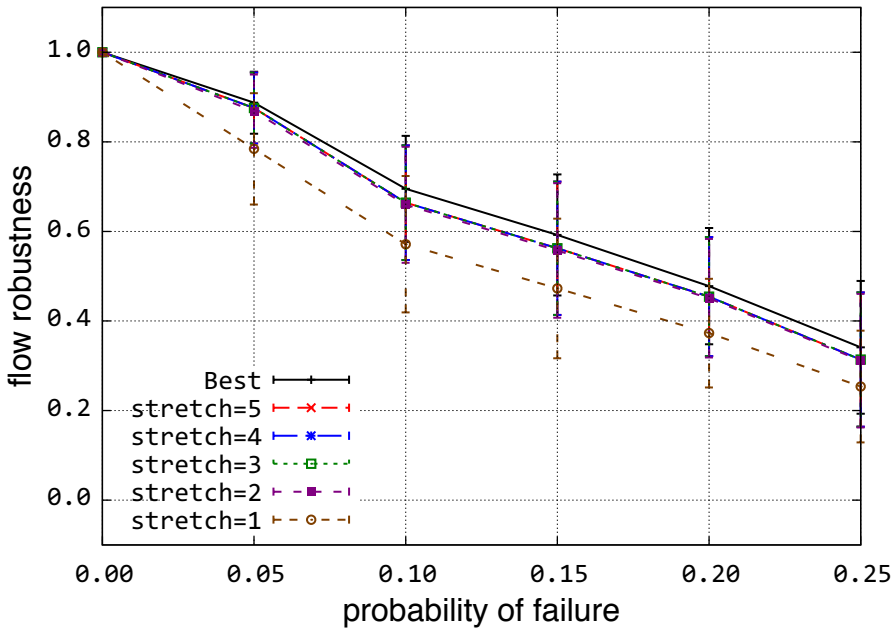

Figure E.23: Flow robustness vs. node failure probability for various stretch limits on the GÉANT2 physical topology

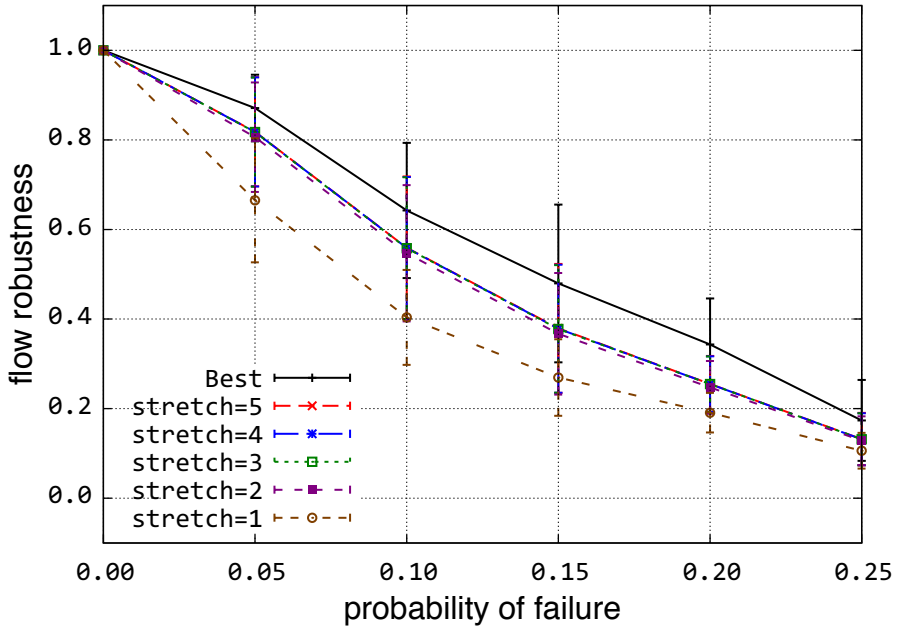

Figure E.24: Flow robustness vs. node & link failure probability for various stretch limits on the GÉANT2 physical topology

E.1.3 Sprint physical

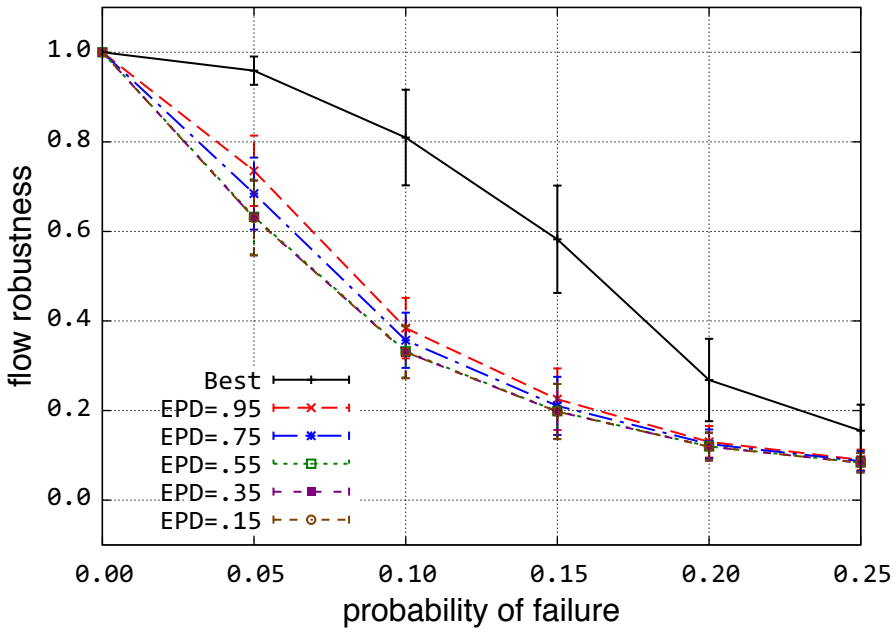

Figure E.25: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Sprint physical topology

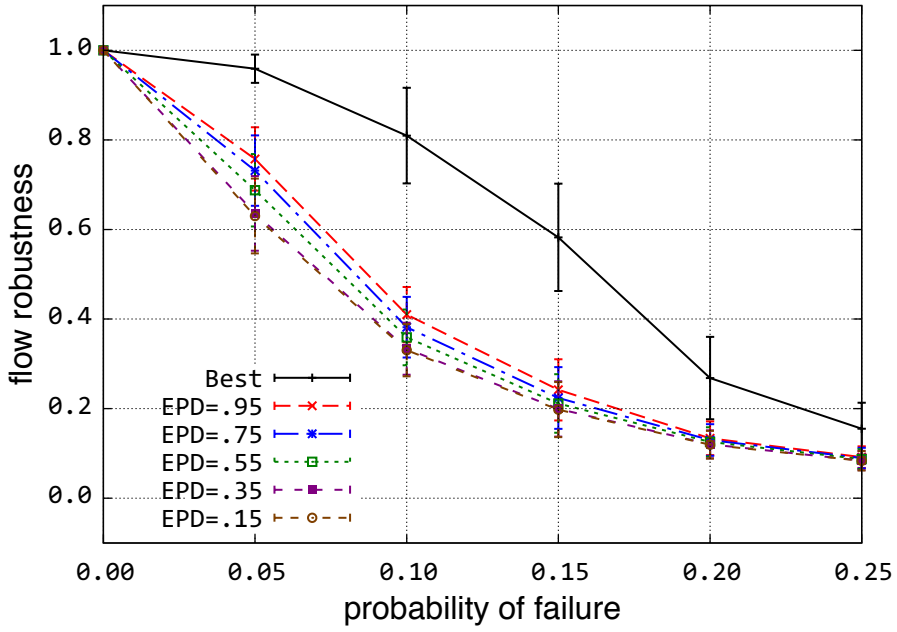

Figure E.26: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Sprint physical topology

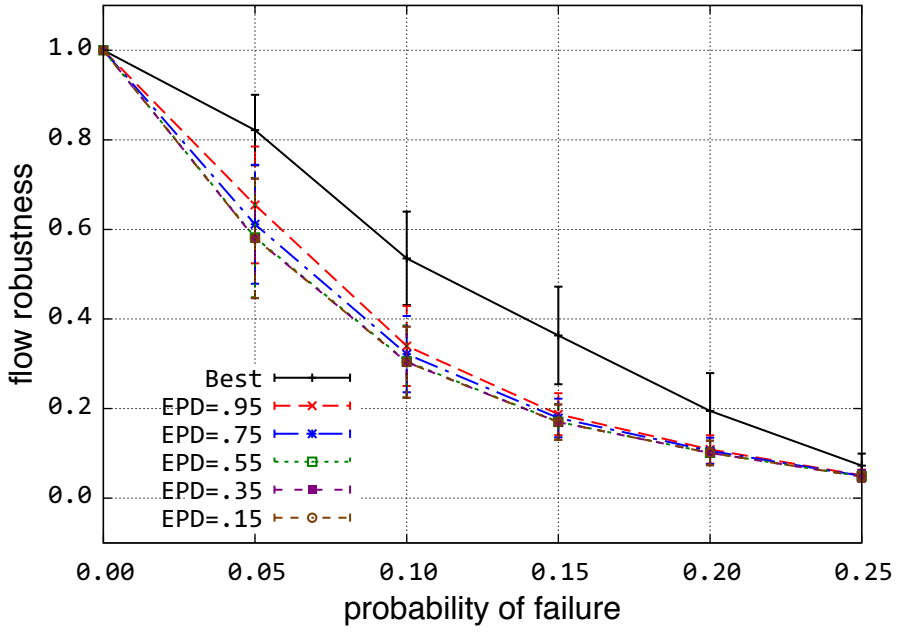

Figure E.27: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Sprint physical topology

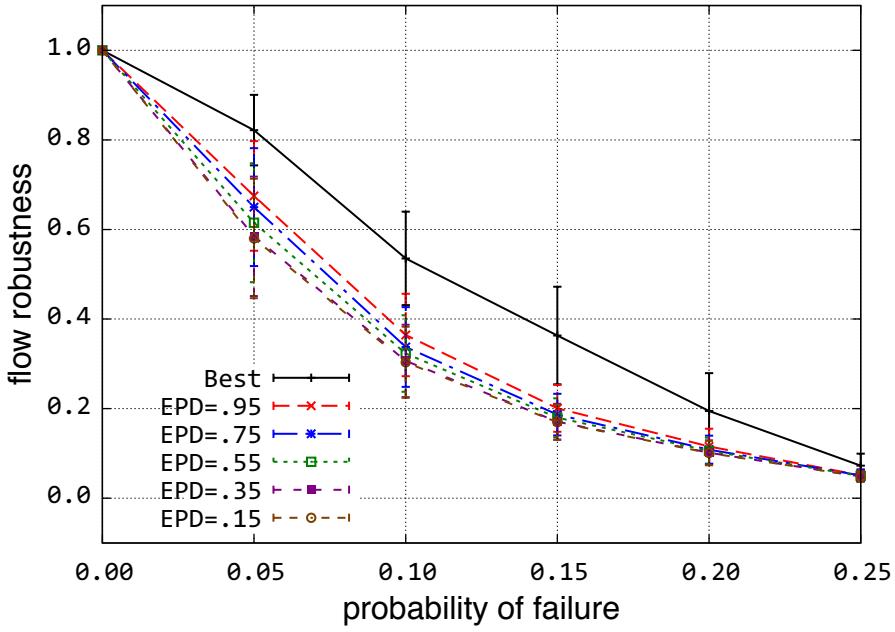

Figure E.28: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Sprint physical topology

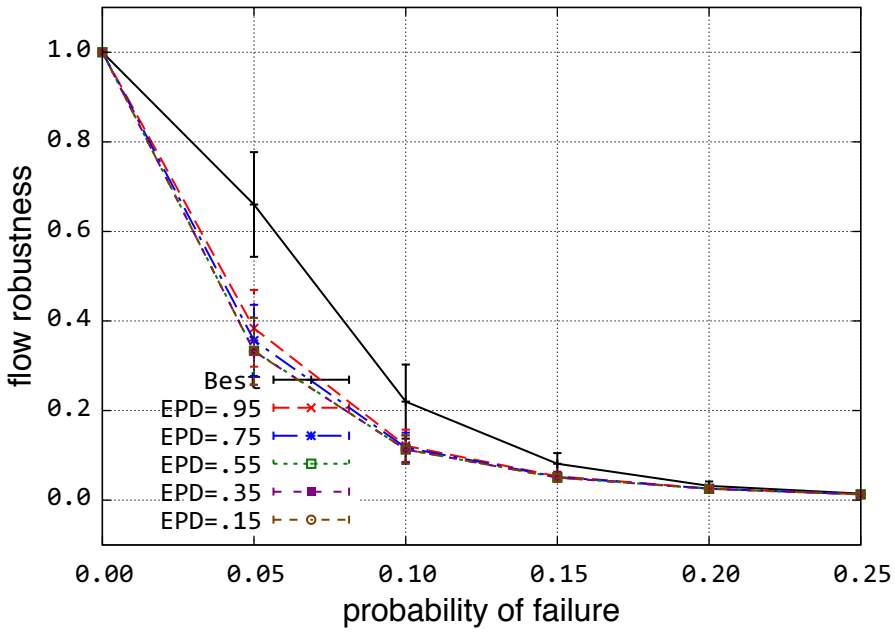

Figure E.29: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Sprint physical topology

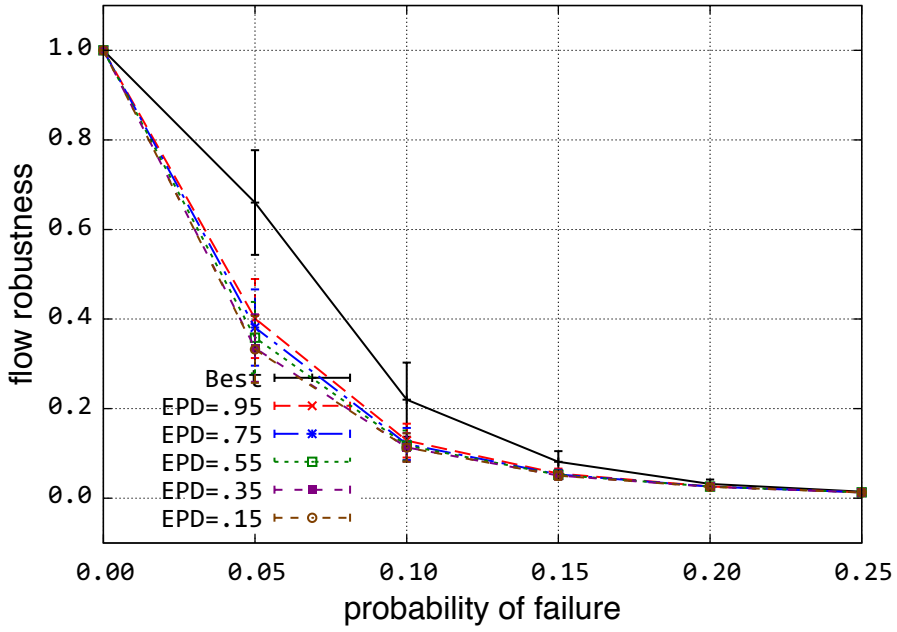

Figure E.30: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Sprint physical topology

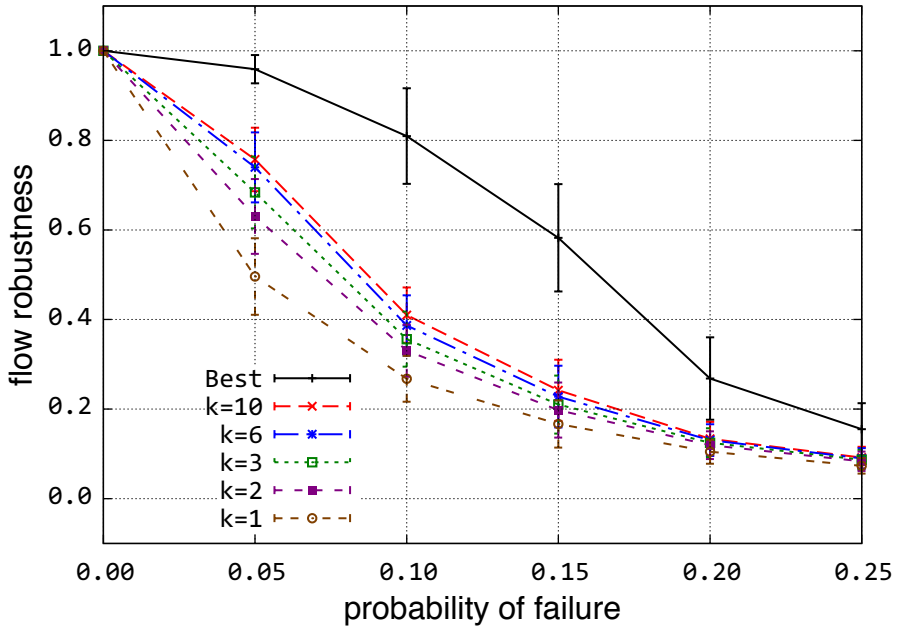

Figure E.31: Flow robustness vs. link failure probability for the Sprint physical topology

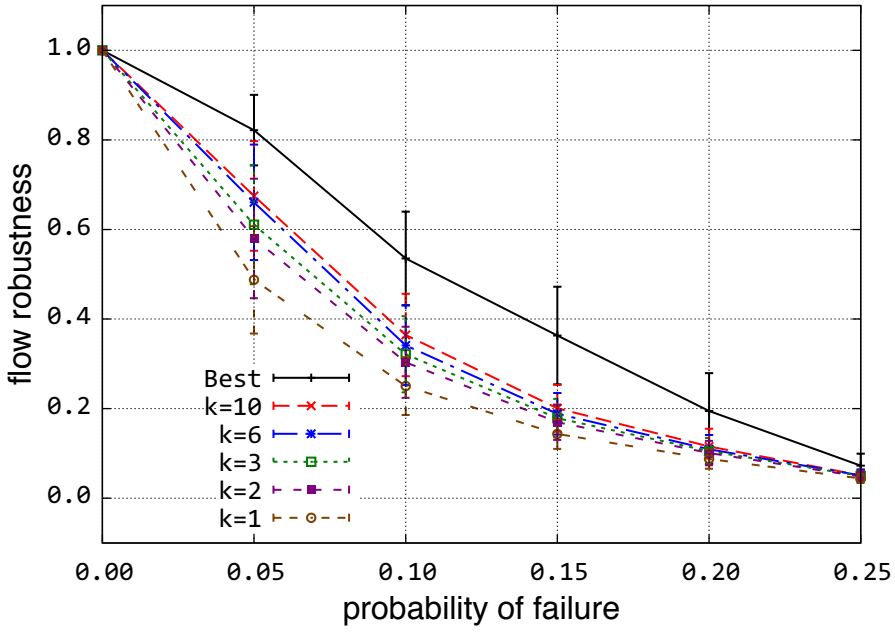

Figure E.32: Flow robustness vs. node failure probability for the Sprint physical topology

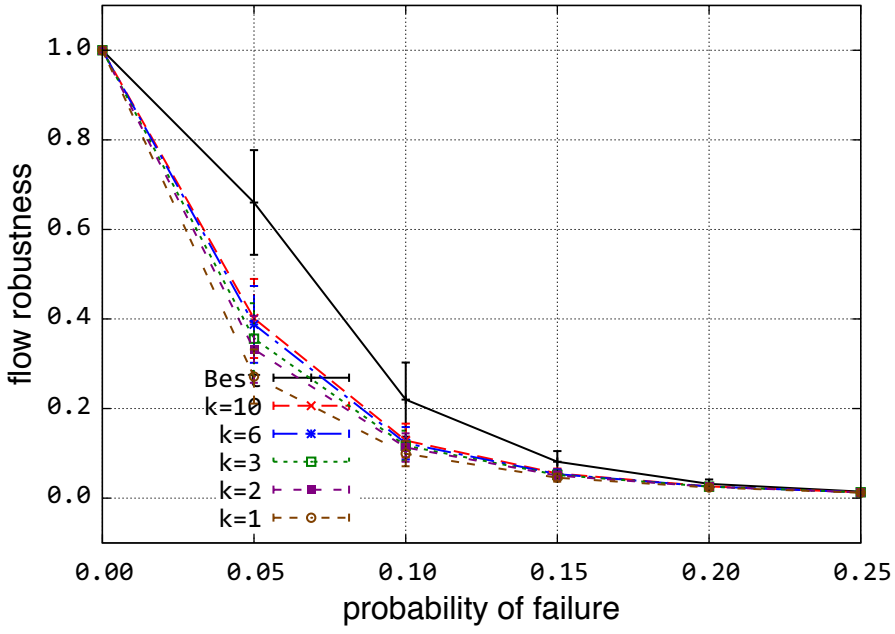

Figure E.33: Flow robustness vs. node & link failure probability for the Sprint physical topology

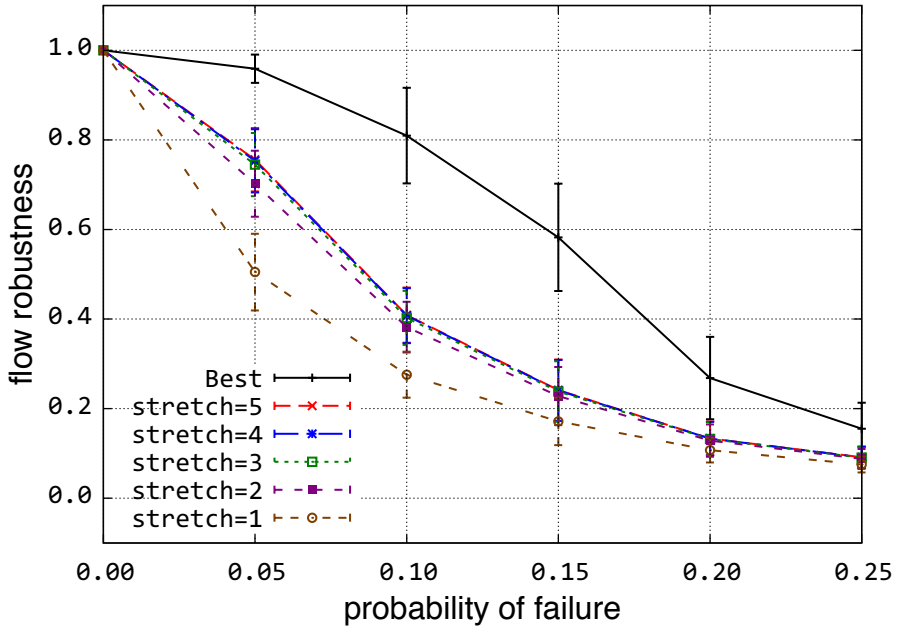

Figure E.34: Flow robustness vs. link failure probability for various stretch limits on the Sprint physical topology

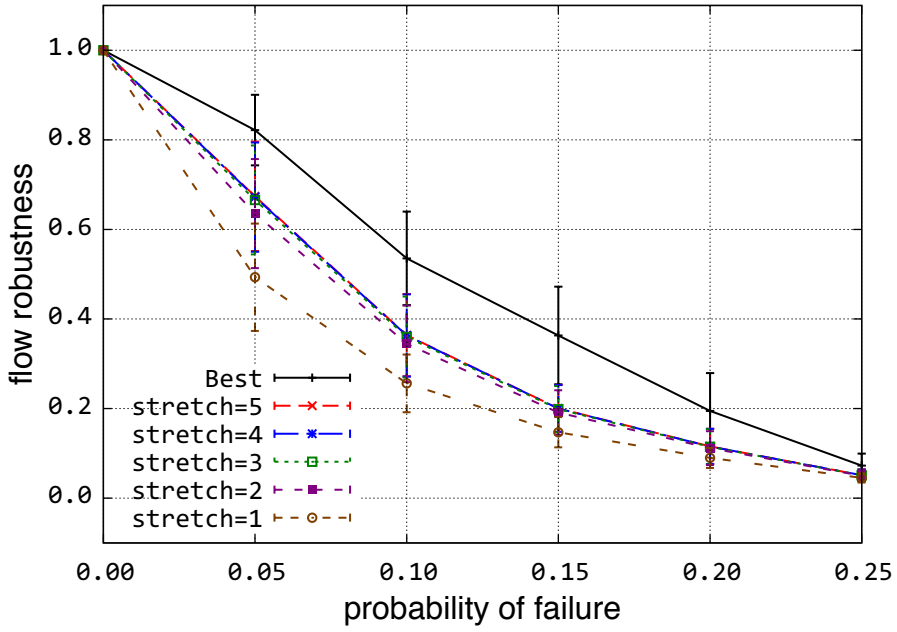

Figure E.35: Flow robustness vs. node failure probability for various stretch limits on the Sprint physical topology

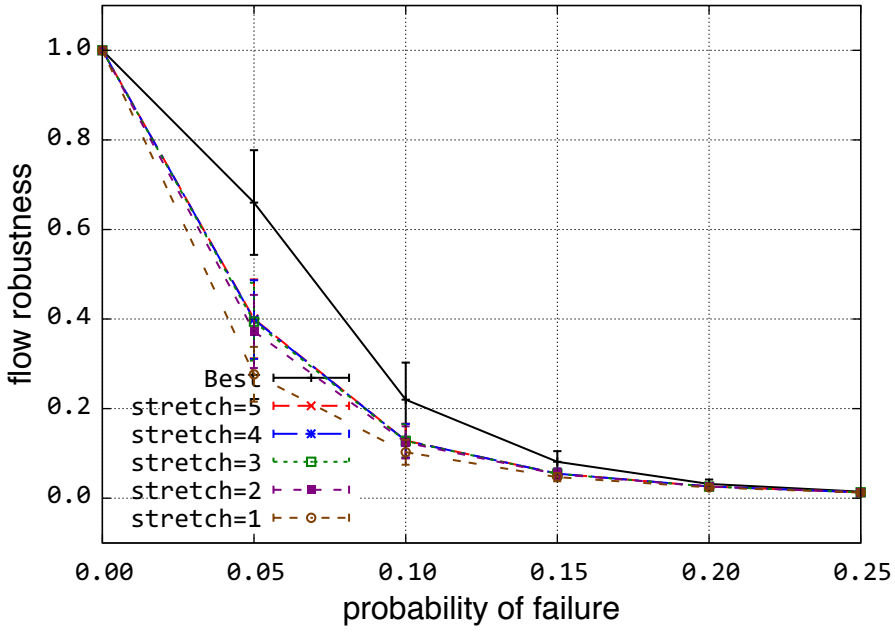

Figure E.36: Flow robustness vs. node & link failure probability for various stretch limits on the Sprint physical topology

E.2 Logical Topologies

E.2.1 AboveNet logical

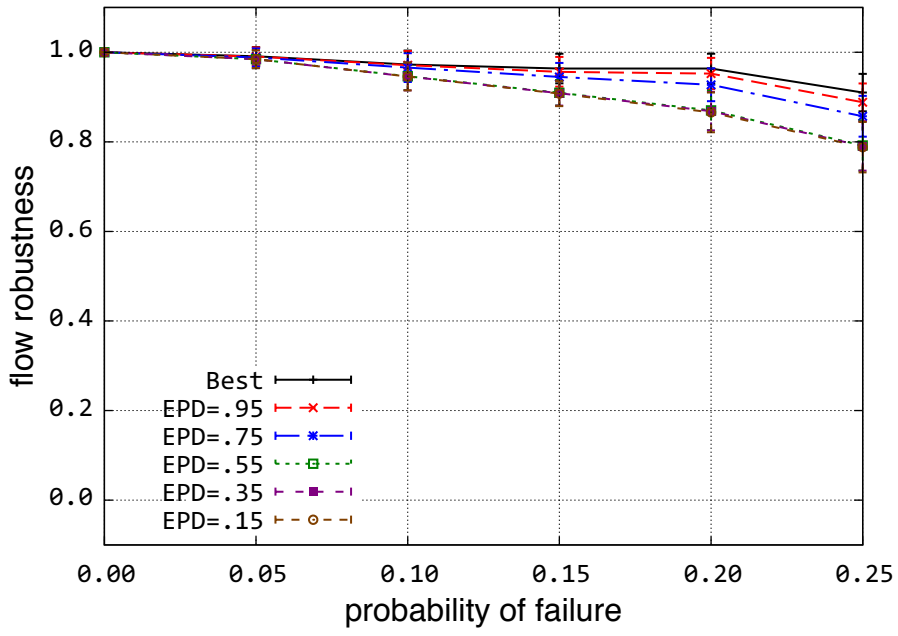

Figure E.37: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the AboveNet logical topology

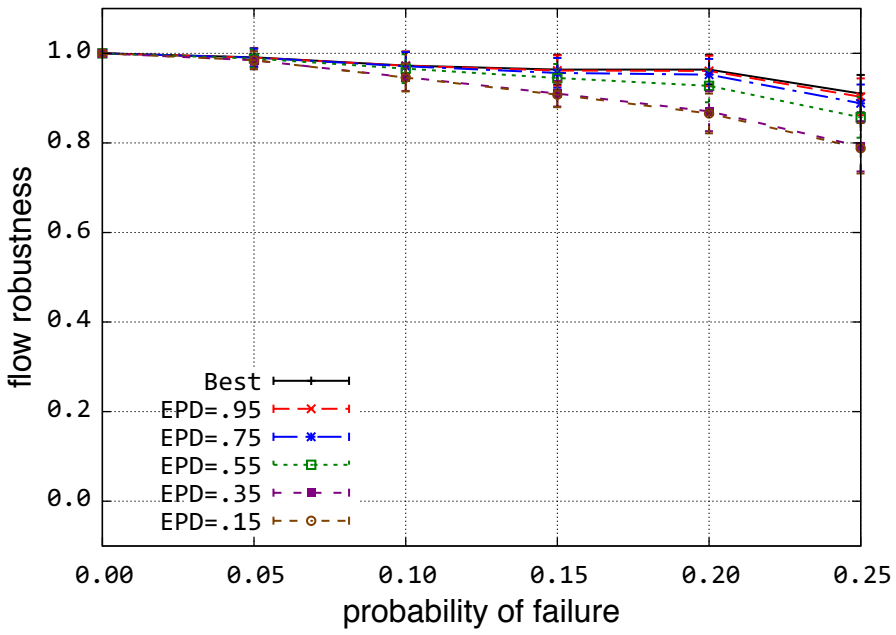

Figure E.38: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the AboveNet logical topology

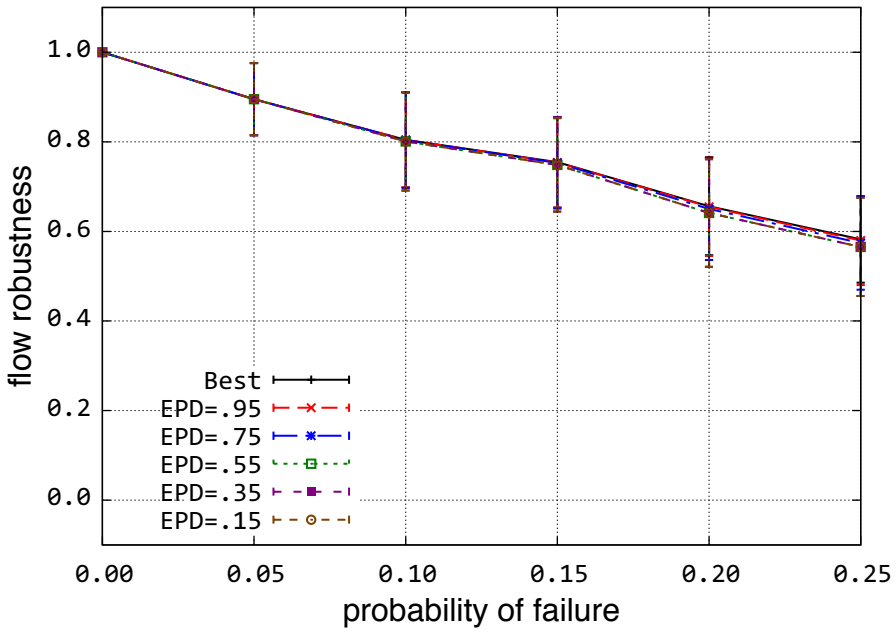

Figure E.39: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the AboveNet logical topology

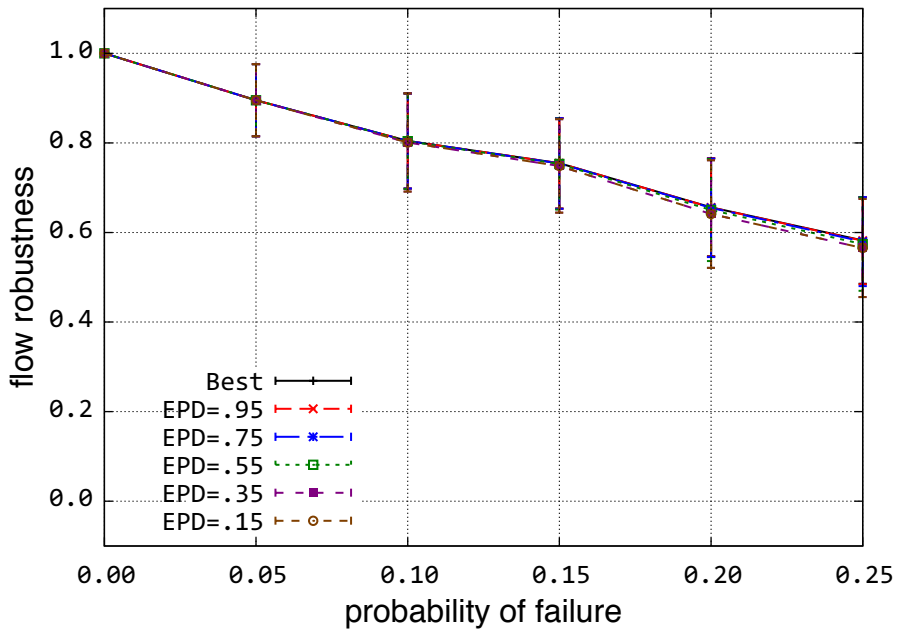

Figure E.40: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the AboveNet logical topology

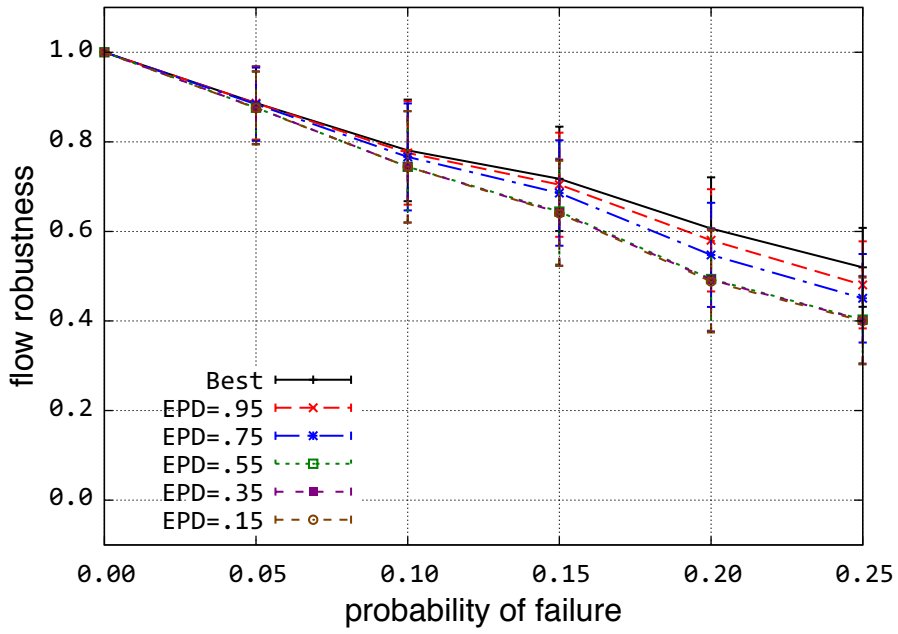

Figure E.41: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the AboveNet logical topology

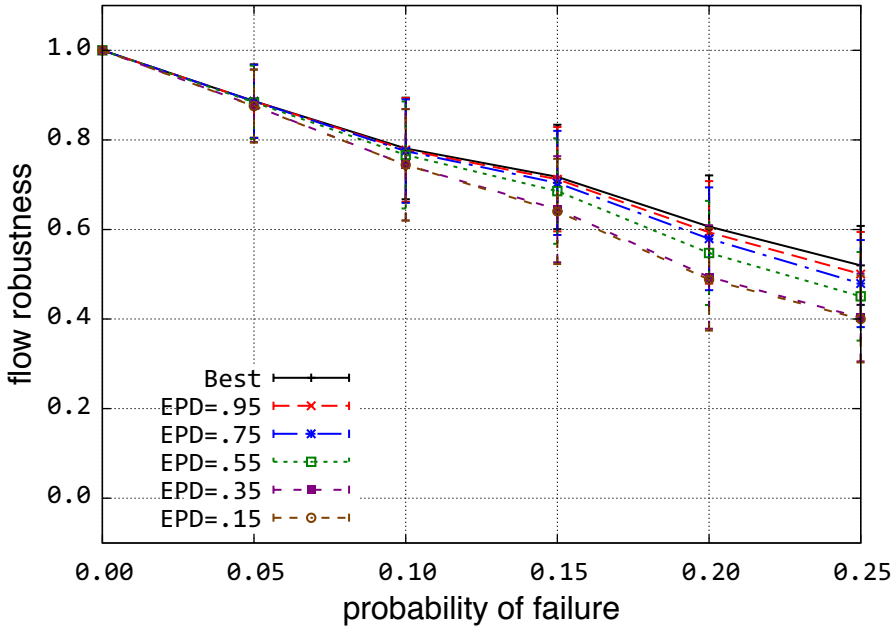

Figure E.42: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the AboveNet logical topology

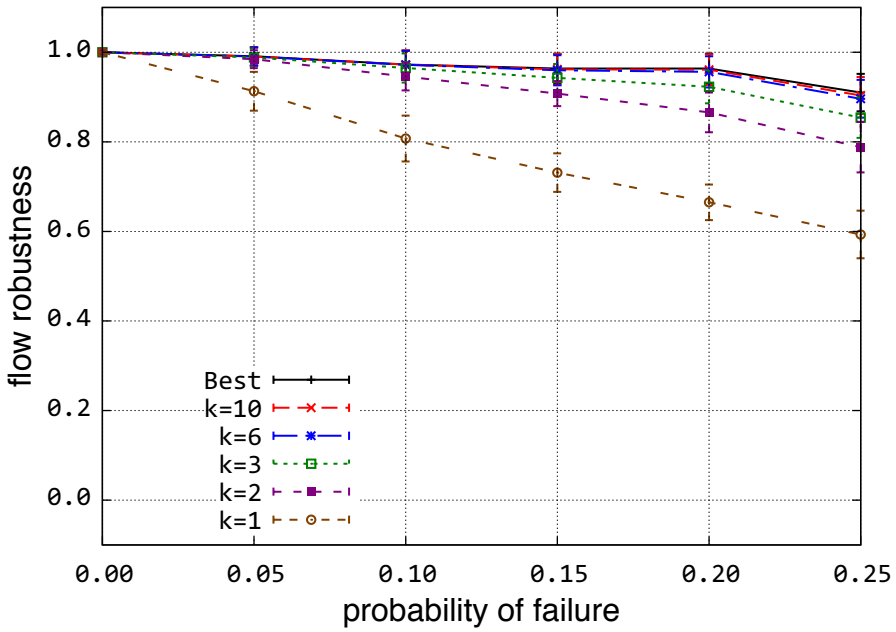

Figure E.43: Flow robustness vs. link failure probability for the AboveNet logical topology

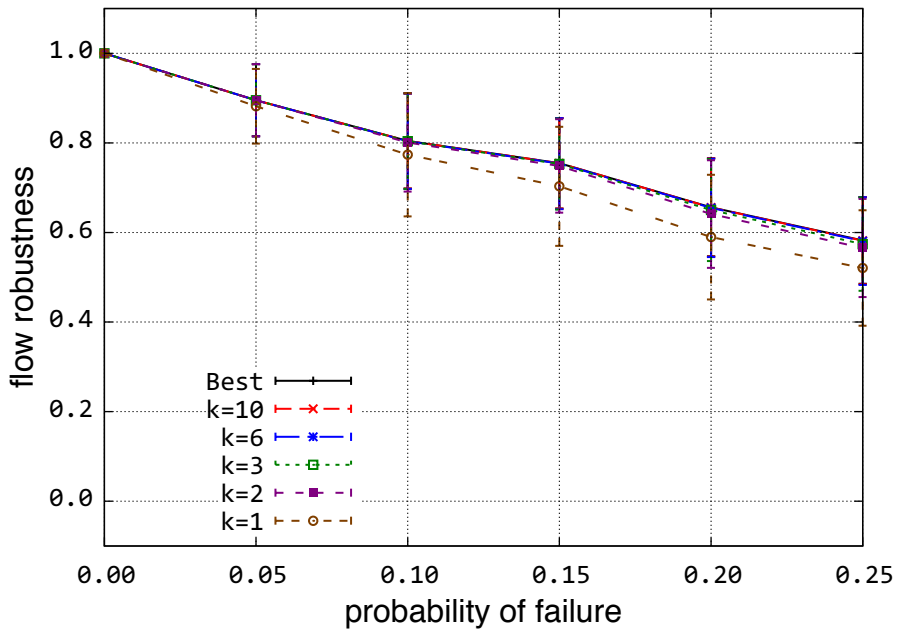

Figure E.44: Flow robustness vs. node failure probability for the AboveNet logical topology

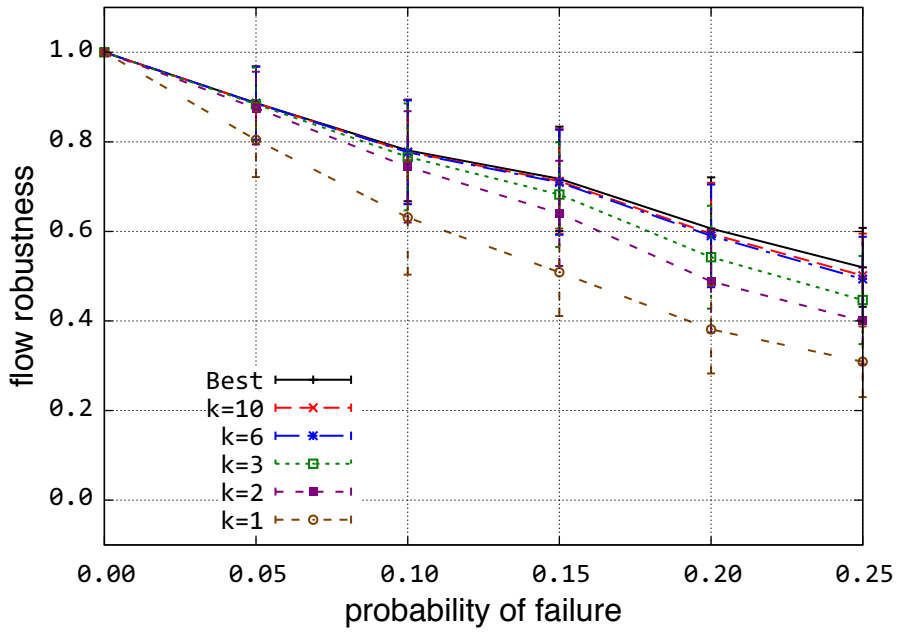

Figure E.45: Flow robustness vs. node & link failure probability for the AboveNet logical topology

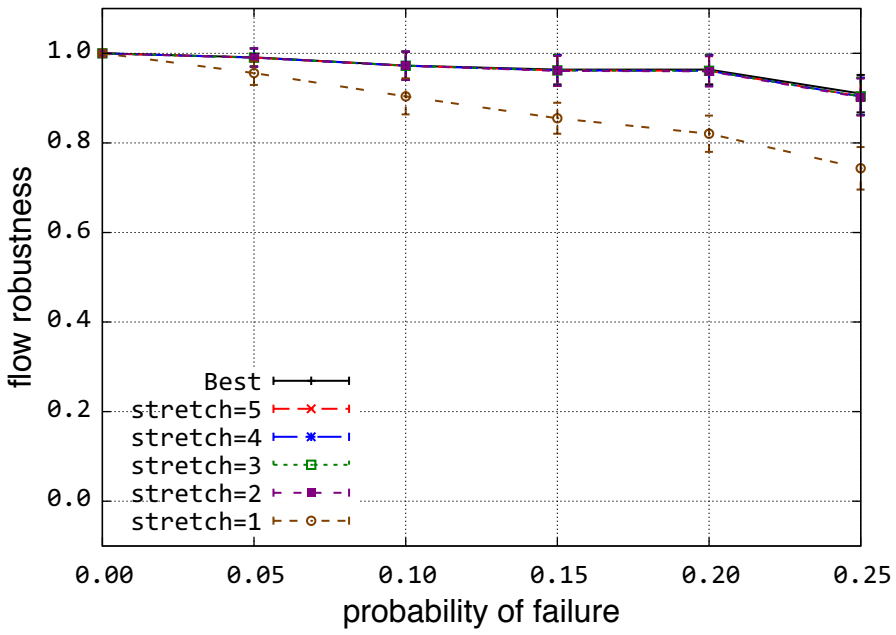

Figure E.46: Flow robustness vs. link failure probability for various stretch limits on the AboveNet logical topology

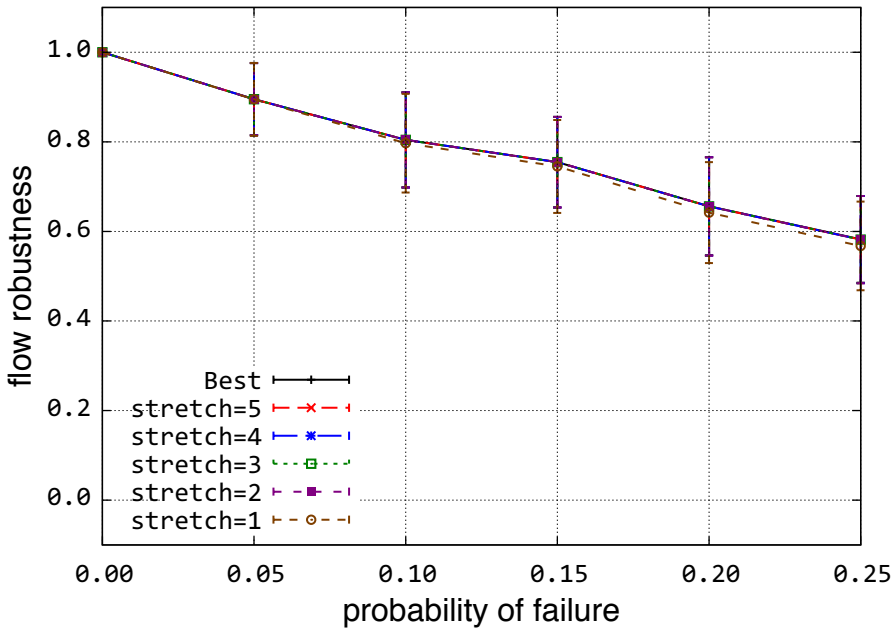

Figure E.47: Flow robustness vs. node failure probability for various stretch limits on the AboveNet logical topology

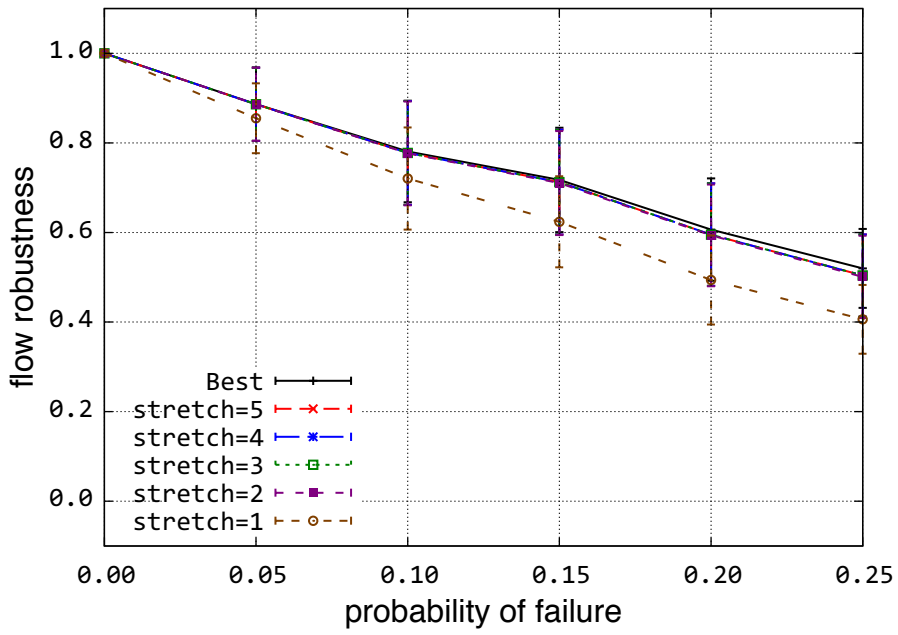

Figure E.48: Flow robustness vs. node & link failure probability for various stretch limits on the AboveNet logical topology

E.2.2 AT&T logical

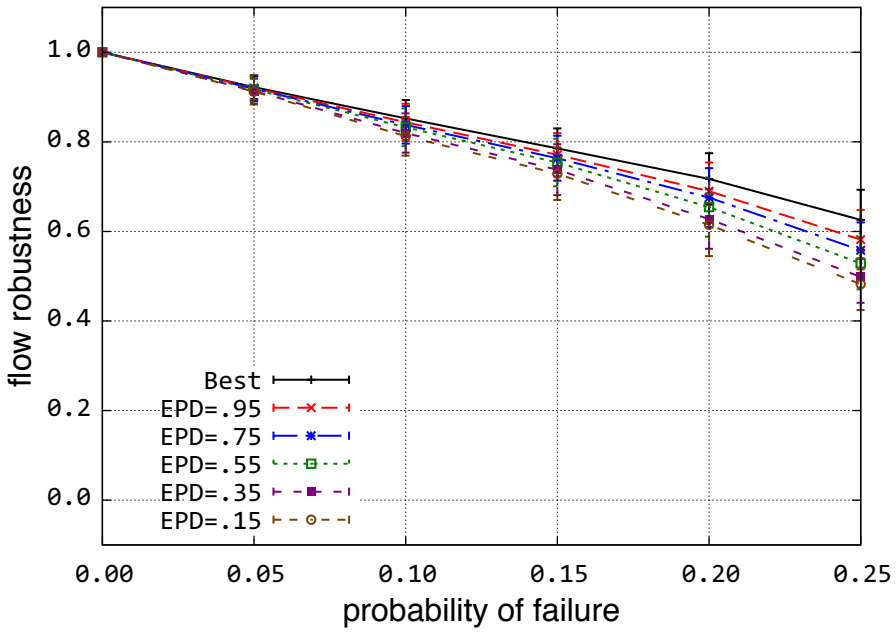

Figure E.49: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the AT&T logical topology

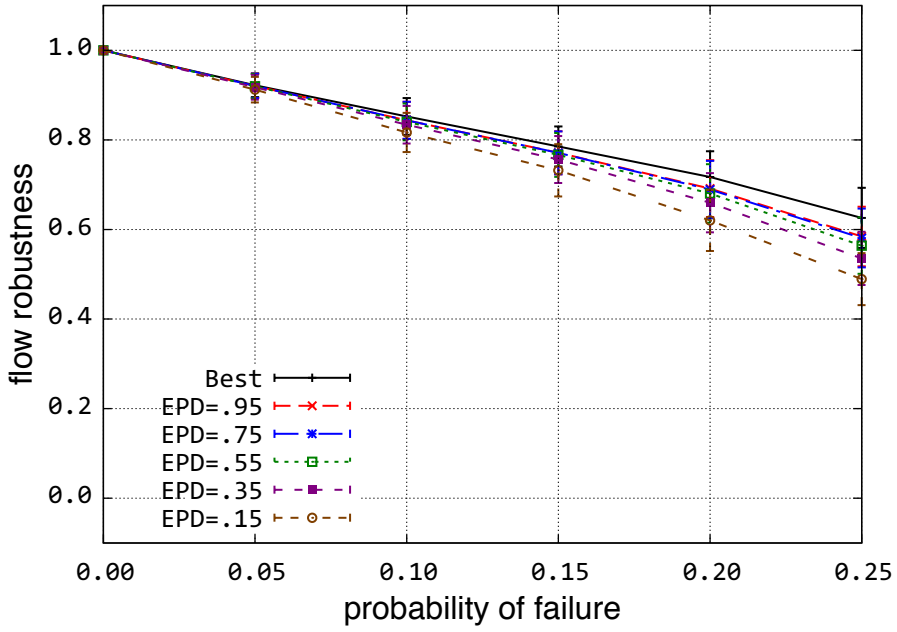

Figure E.50: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the AT&T logical topology

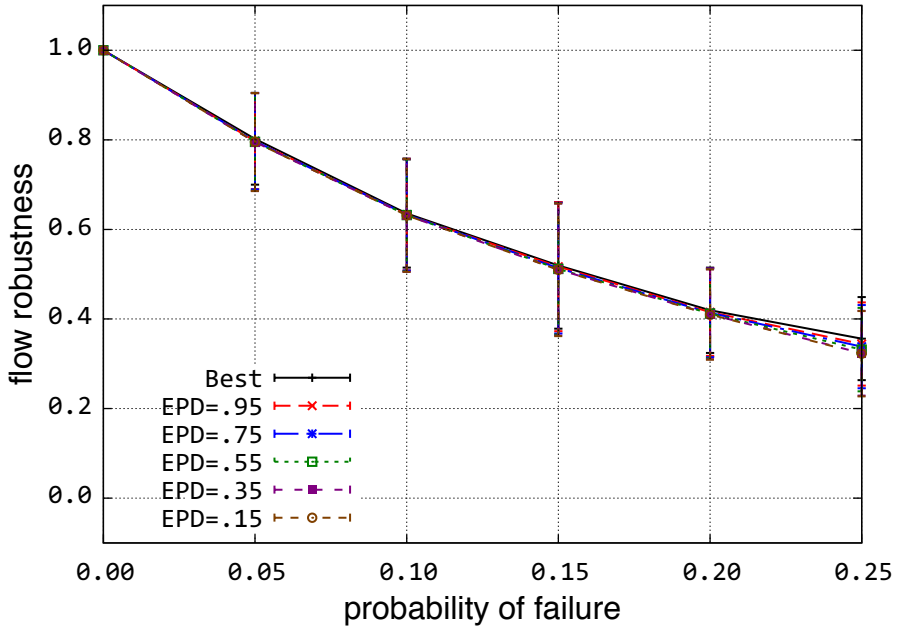

Figure E.51: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the AT&T logical topology

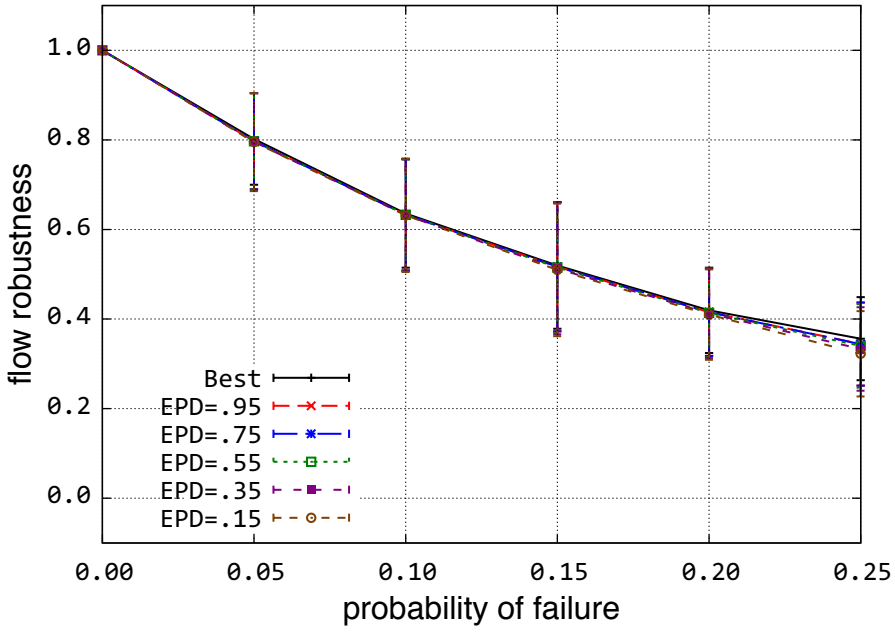

Figure E.52: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the AT&T logical topology

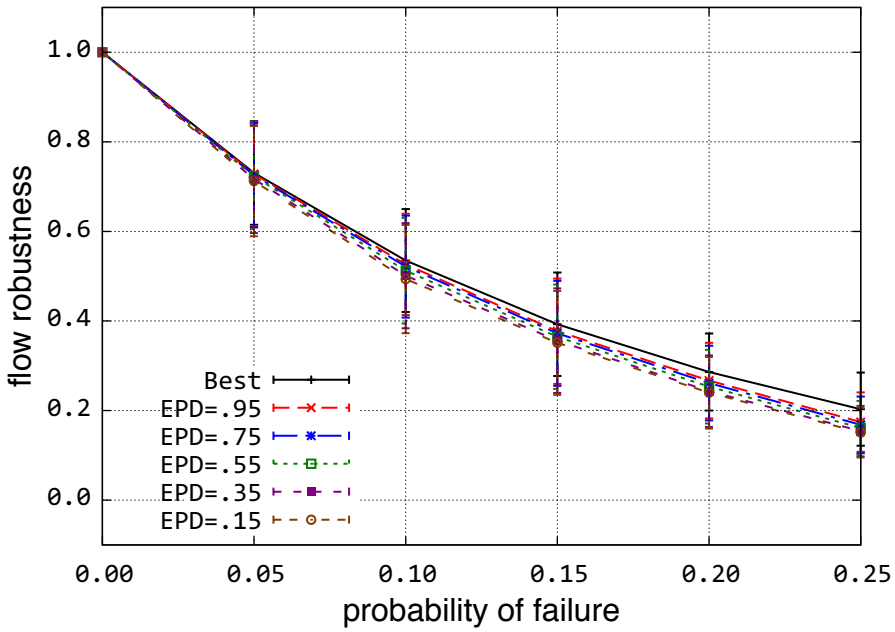

Figure E.53: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the AT&T logical topology

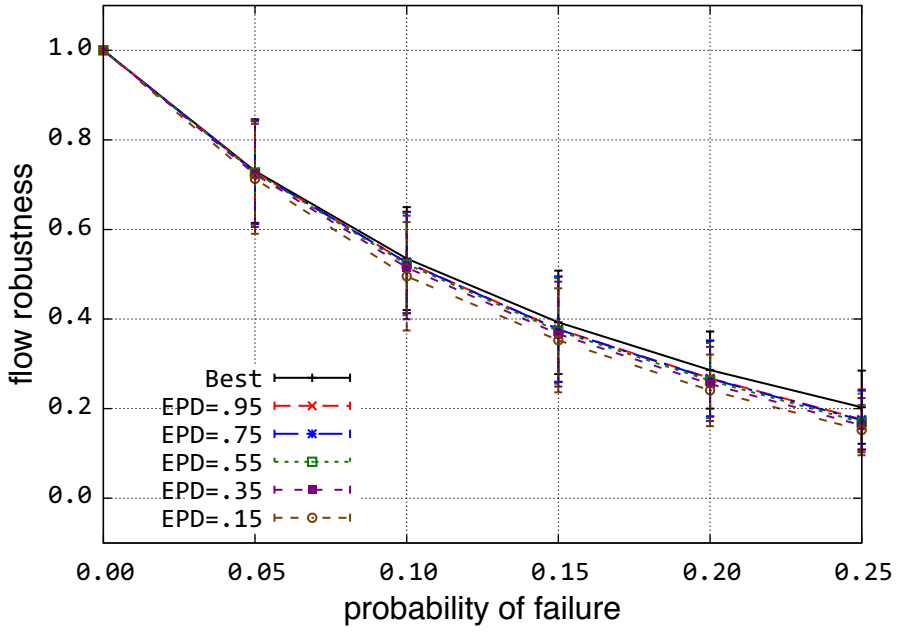

Figure E.54: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the AT&T logical topology

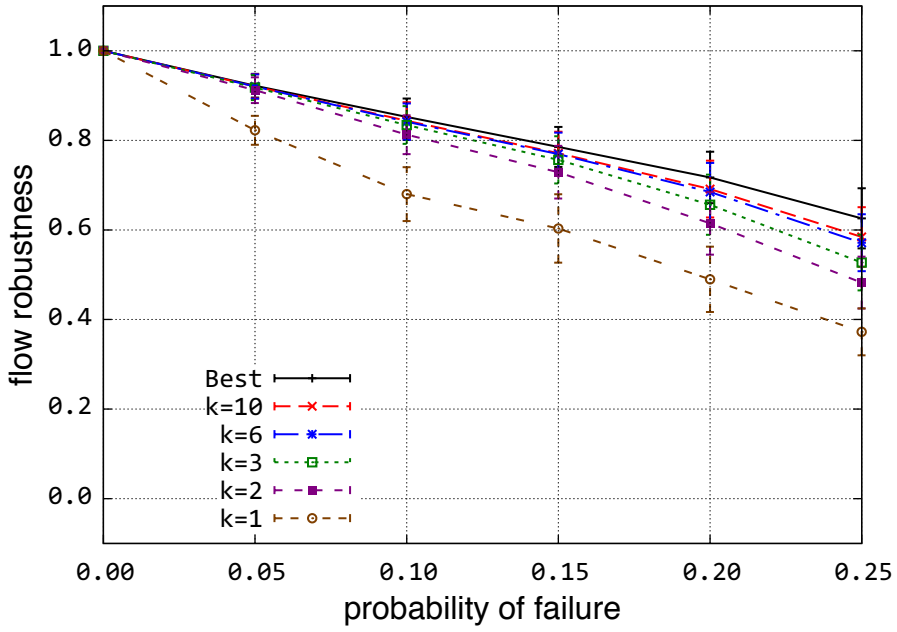

Figure E.55: Flow robustness vs. link failure probability for the AT&T logical topology

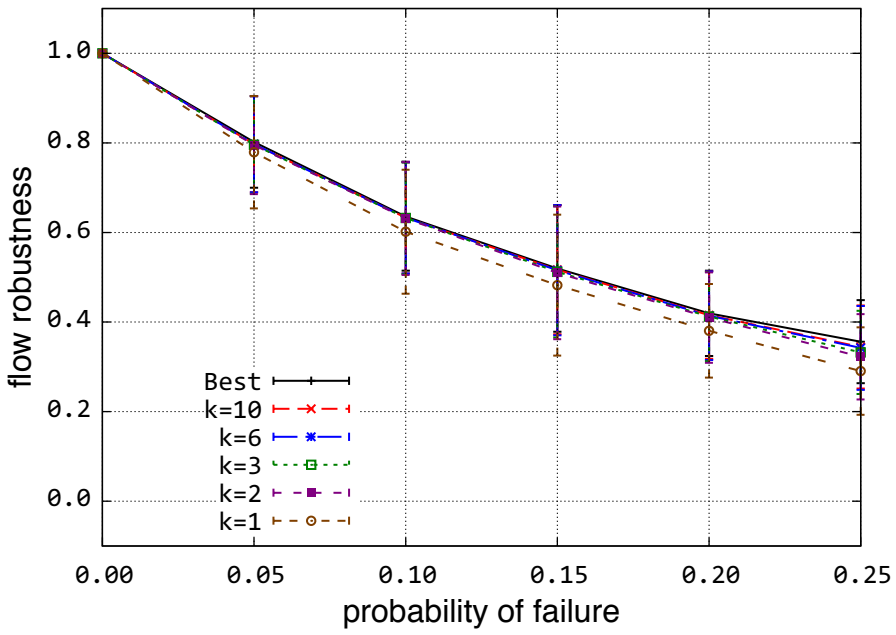

Figure E.56: Flow robustness vs. node failure probability for the AT&T logical topology

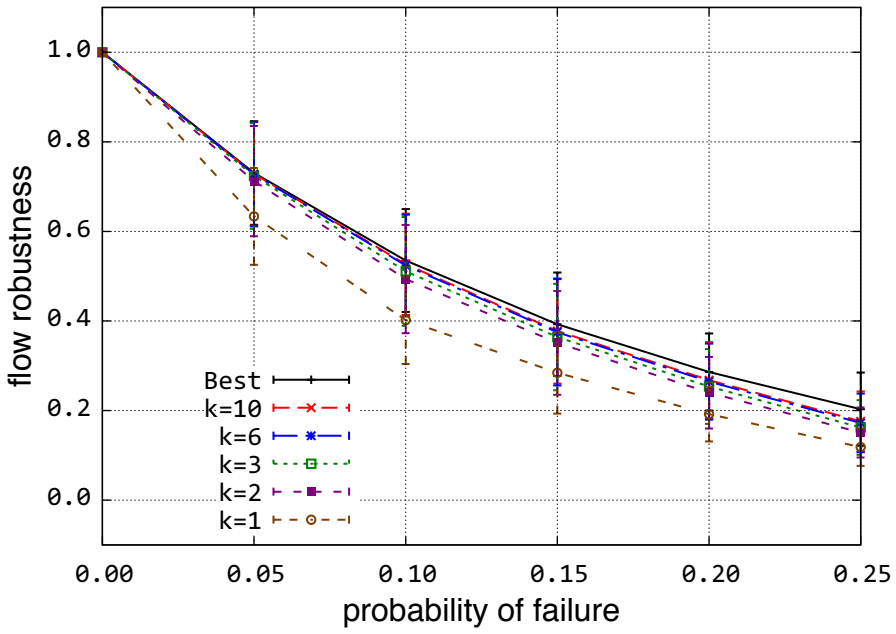

Figure E.57: Flow robustness vs. node & link failure probability for the AT&T logical topology

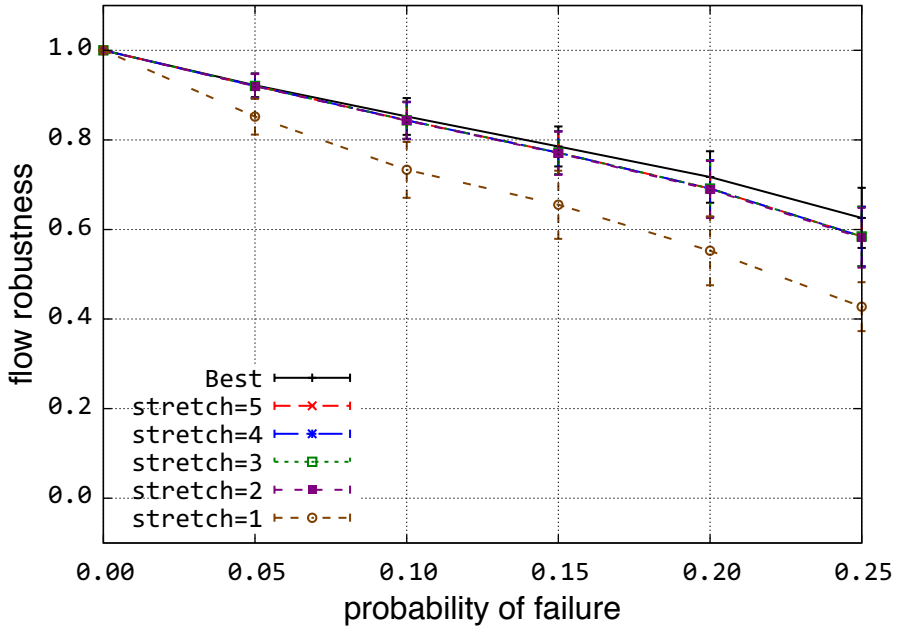

Figure E.58: Flow robustness vs. link failure probability for various stretch limits on the AT&T logical topology

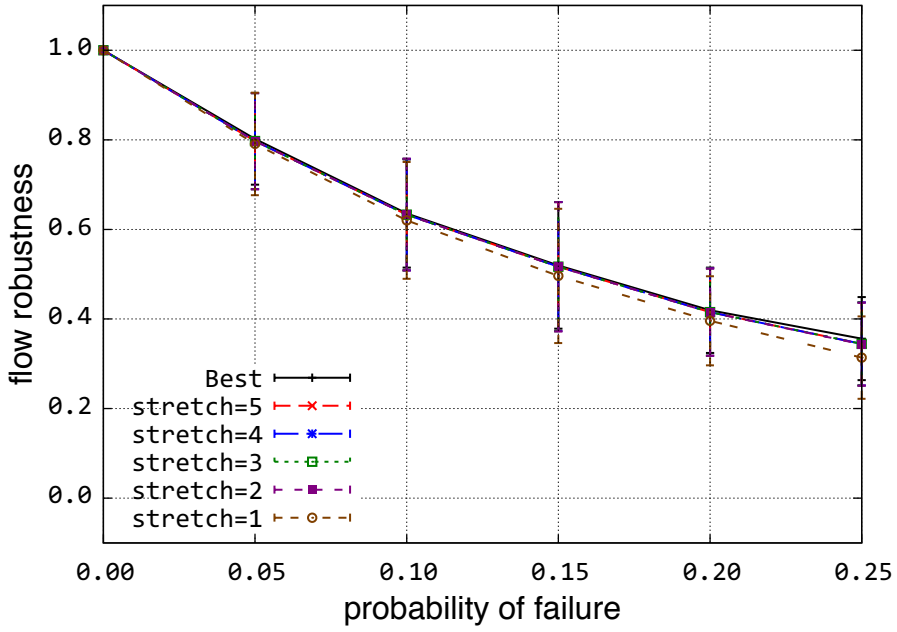

Figure E.59: Flow robustness vs. node failure probability for various stretch limits on the AT&T logical topology

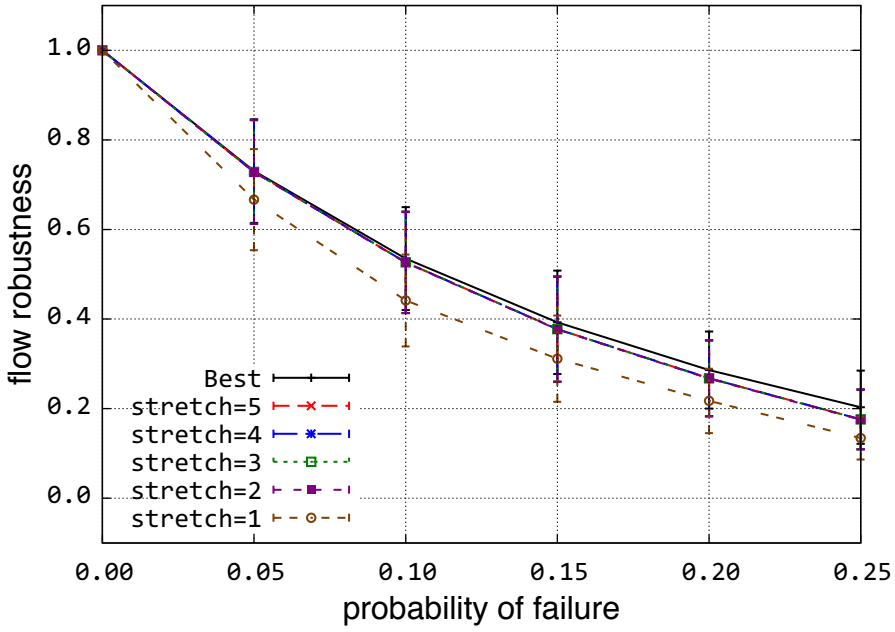

Figure E.60: Flow robustness vs. node & link failure probability for various stretch limits on the AT&T logical topology

E.2.3 EBONE logical

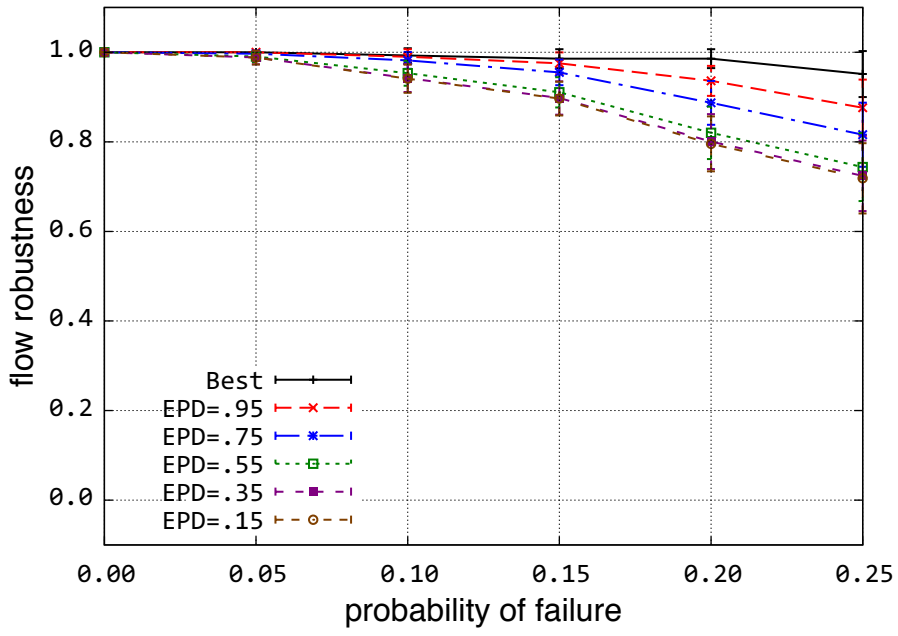

Figure E.61: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the EBONE logical topology

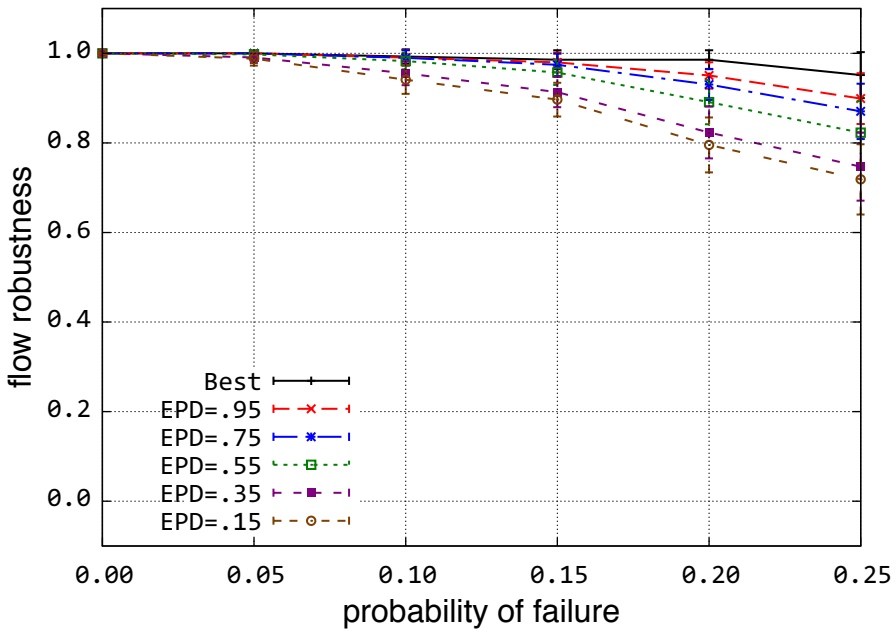

Figure E.62: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the EBONE logical topology

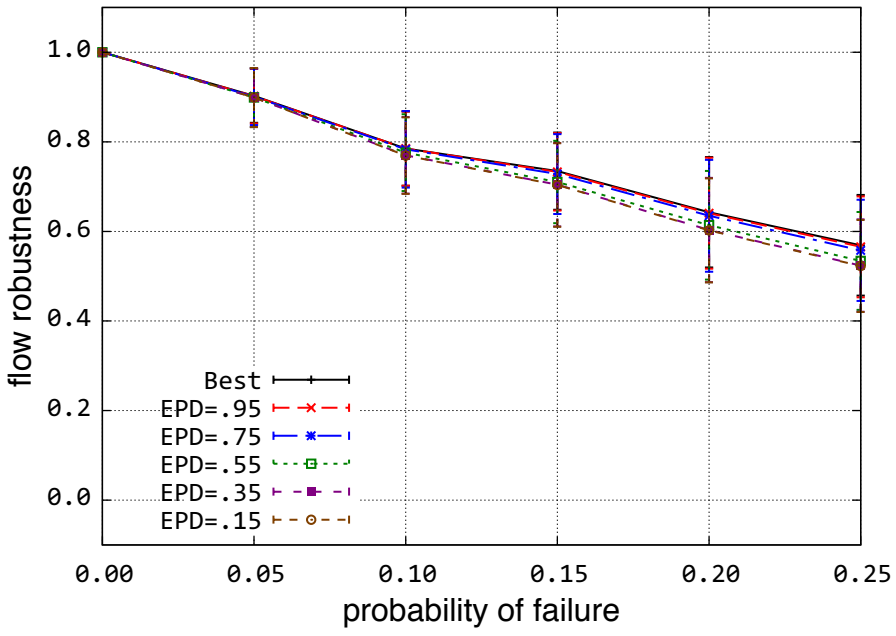

Figure E.63: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the EBONE logical topology

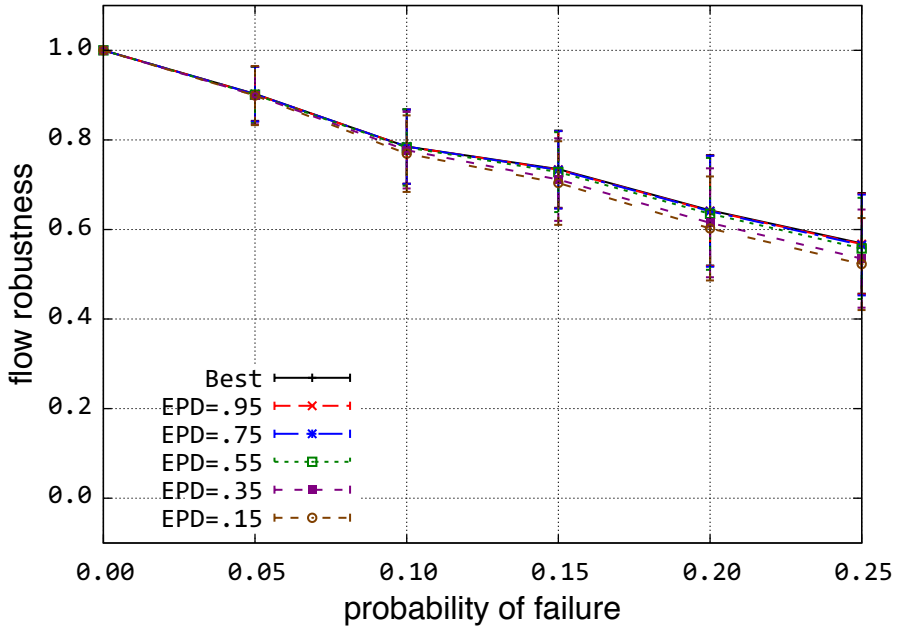

Figure E.64: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the EBONE logical topology

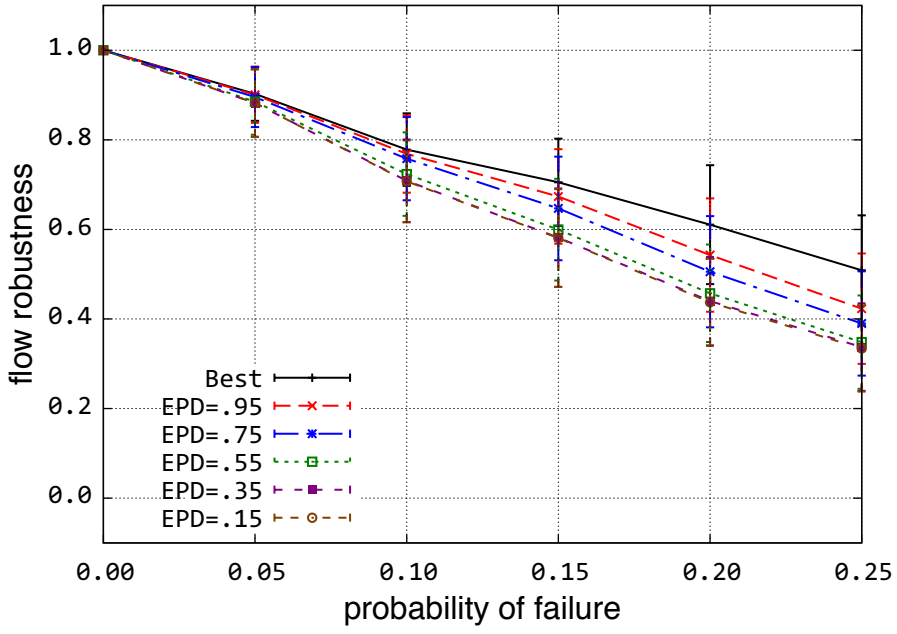

Figure E.65: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the EBONE logical topology

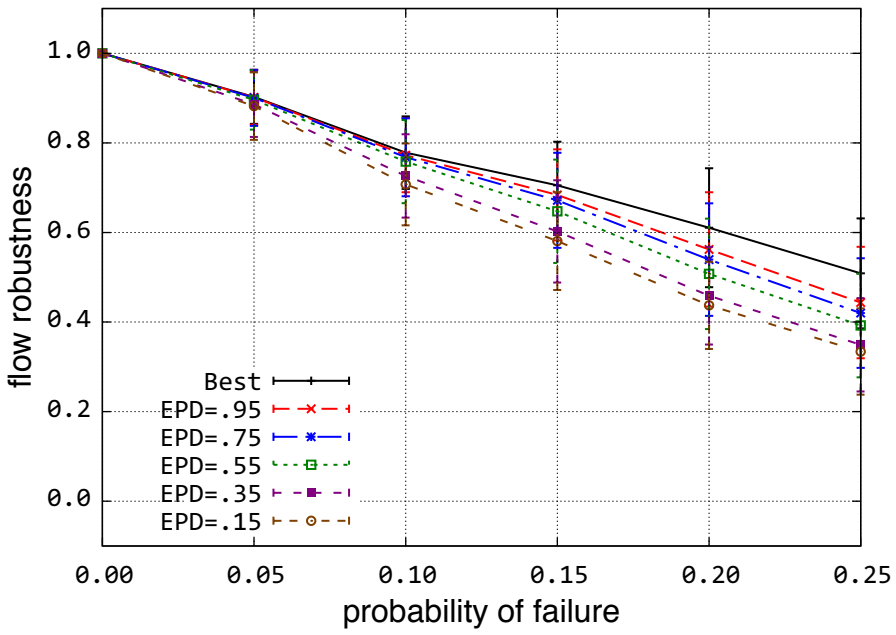

Figure E.66: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the EBONE logical topology

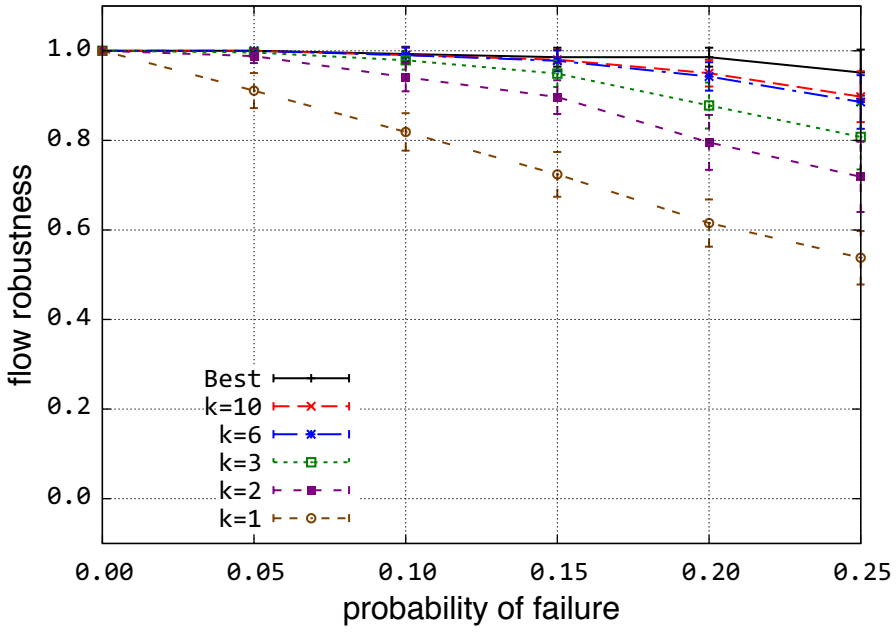

Figure E.67: Flow robustness vs. link failure probability for the EBONE logical topology

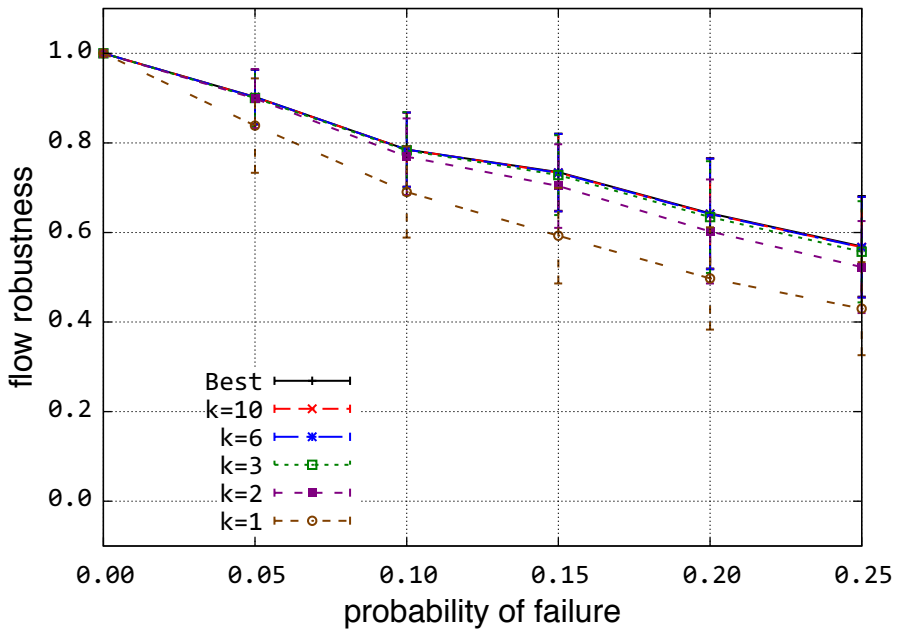

Figure E.68: Flow robustness vs. node failure probability for the EBONE logical topology

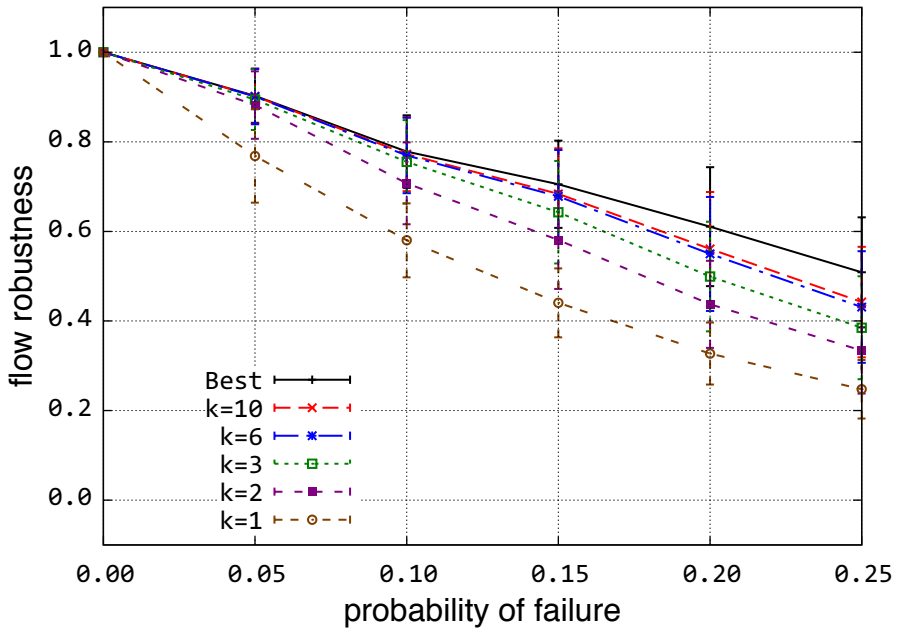

Figure E.69: Flow robustness vs. node & link failure probability for the EBONE logical topology

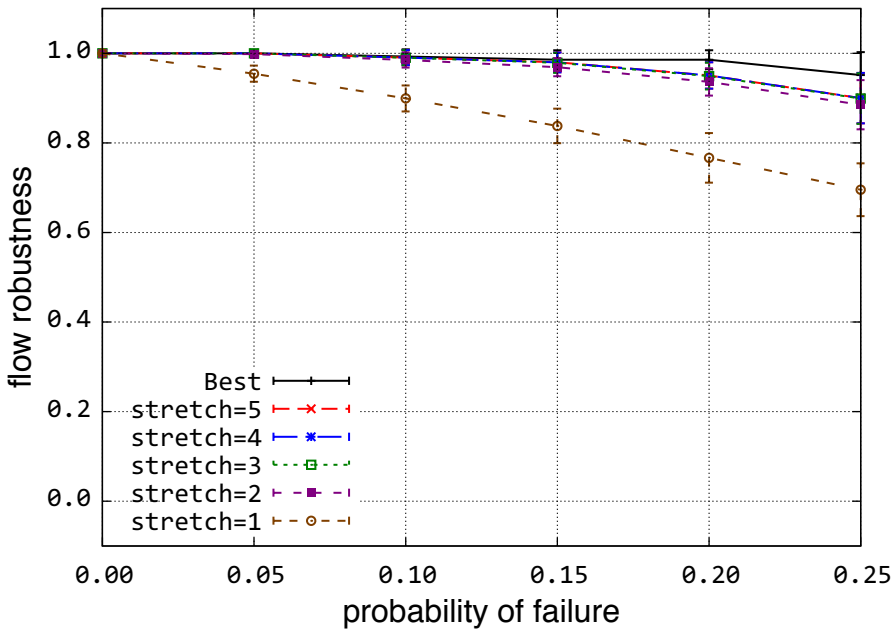

Figure E.70: Flow robustness vs. link failure probability for various stretch limits on the EBONE logical topology

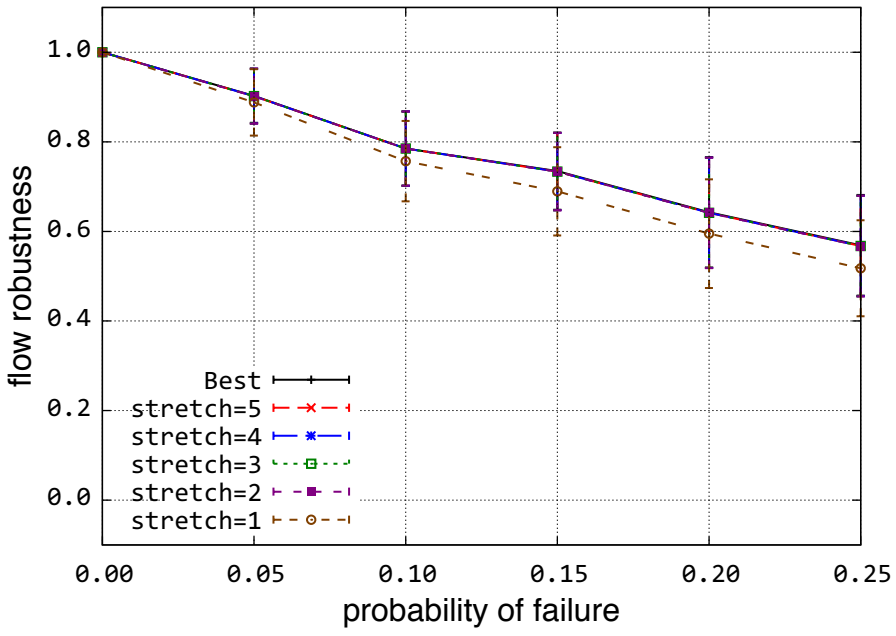

Figure E.71: Flow robustness vs. node failure probability for various stretch limits on the EBONE logical topology

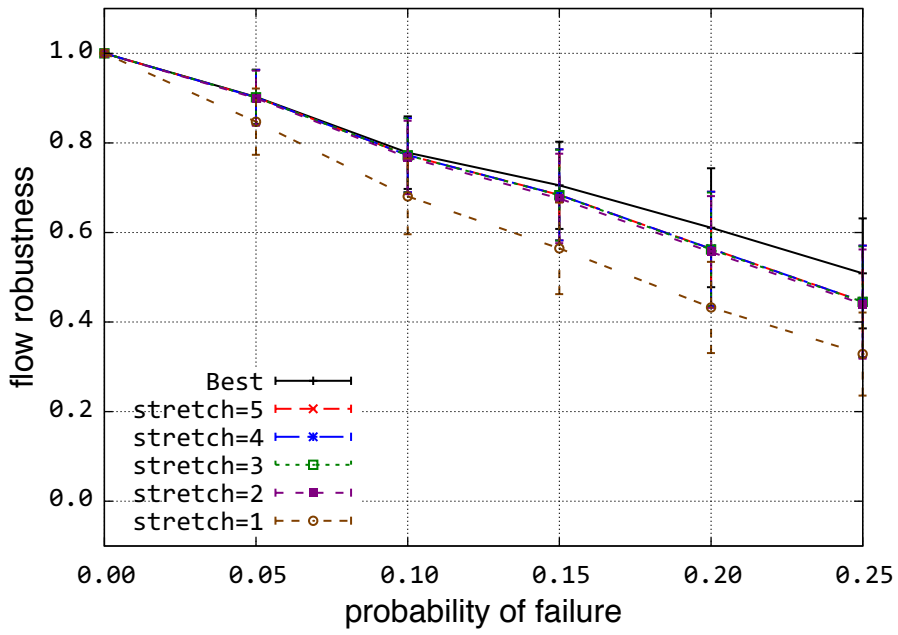

Figure E.72: Flow robustness vs. node & link failure probability for various stretch limits on the EBONE logical topology

E.2.4 Exodus logical

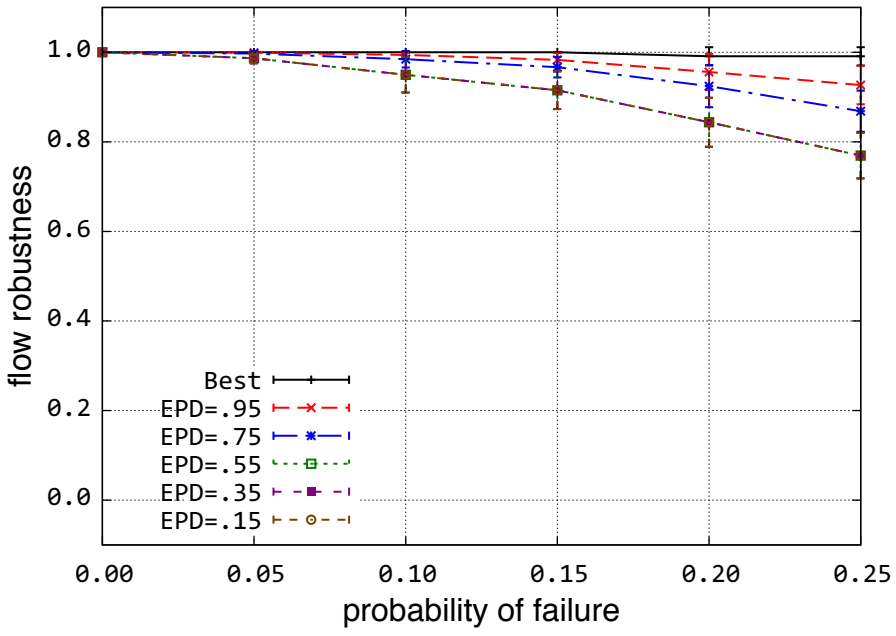

Figure E.73: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Exodus logical topology

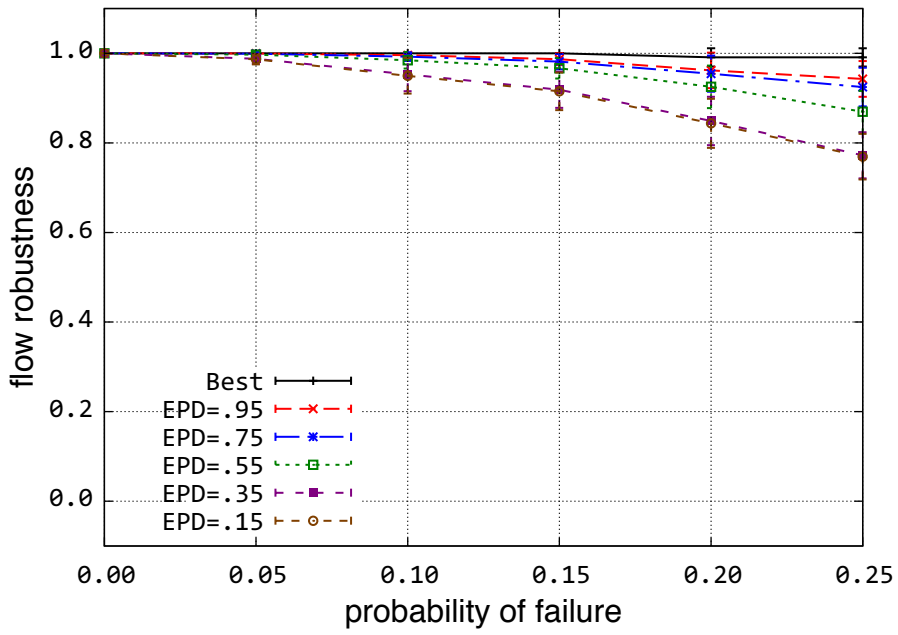

Figure E.74: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Exodus logical topology

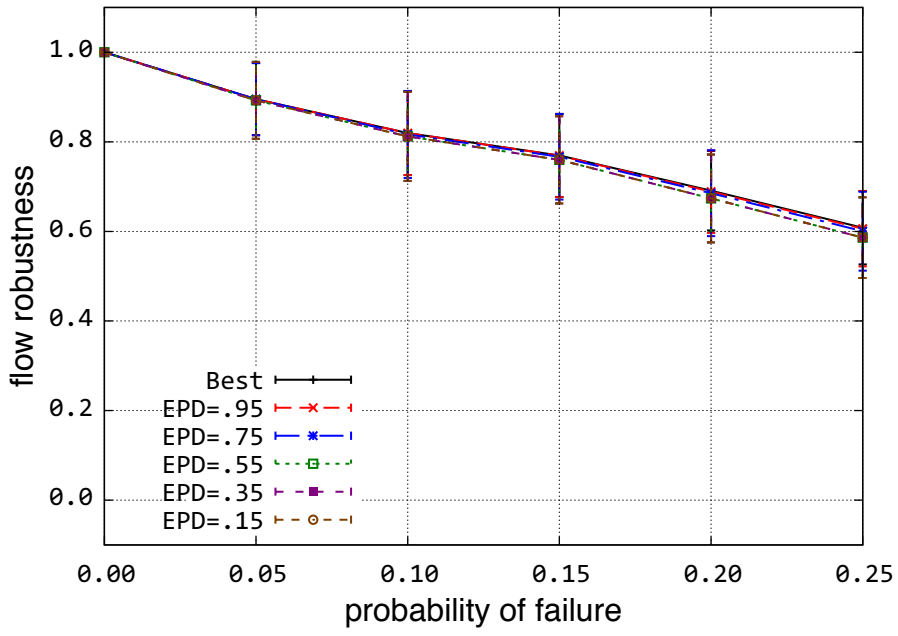

Figure E.75: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Exodus logical topology

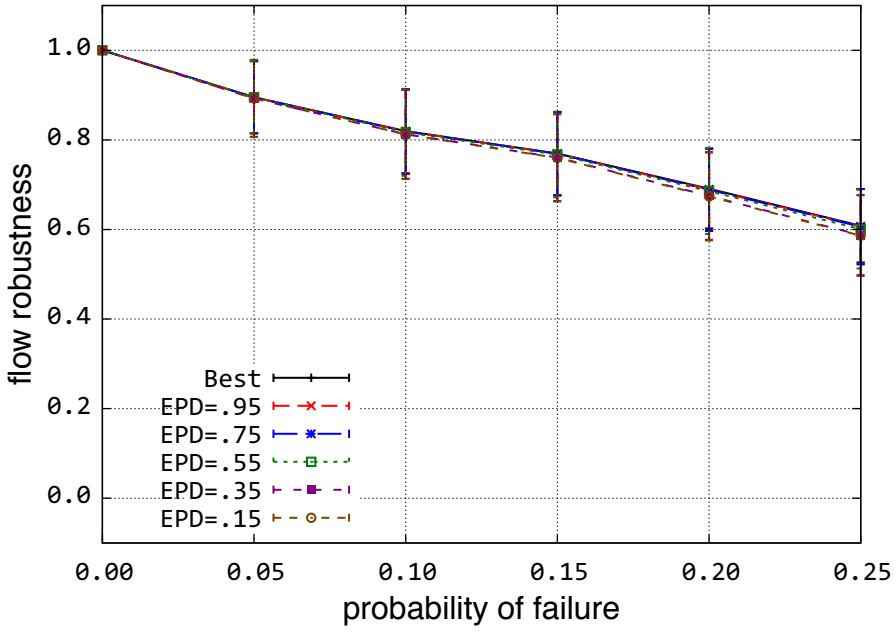

Figure E.76: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Exodus logical topology

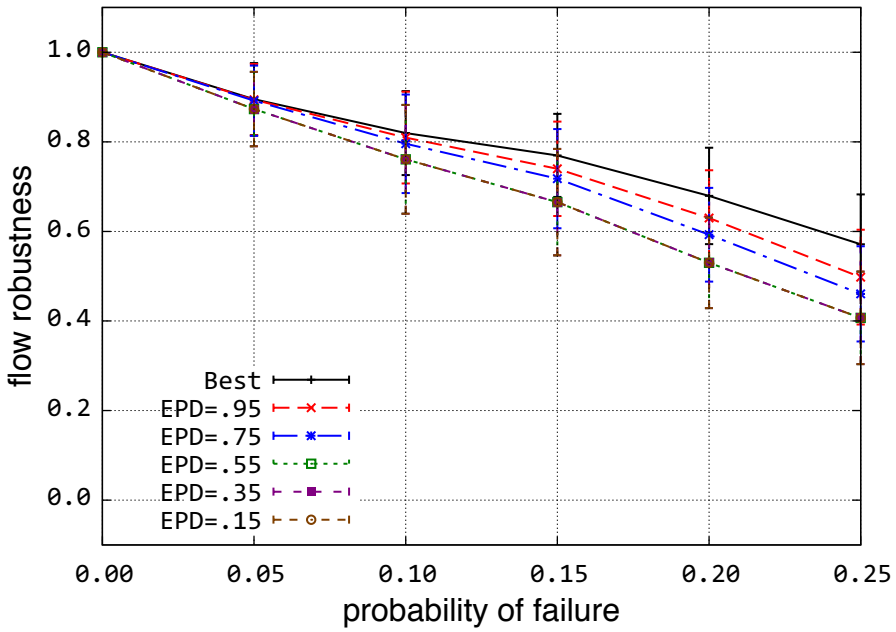

Figure E.77: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Exodus logical topology

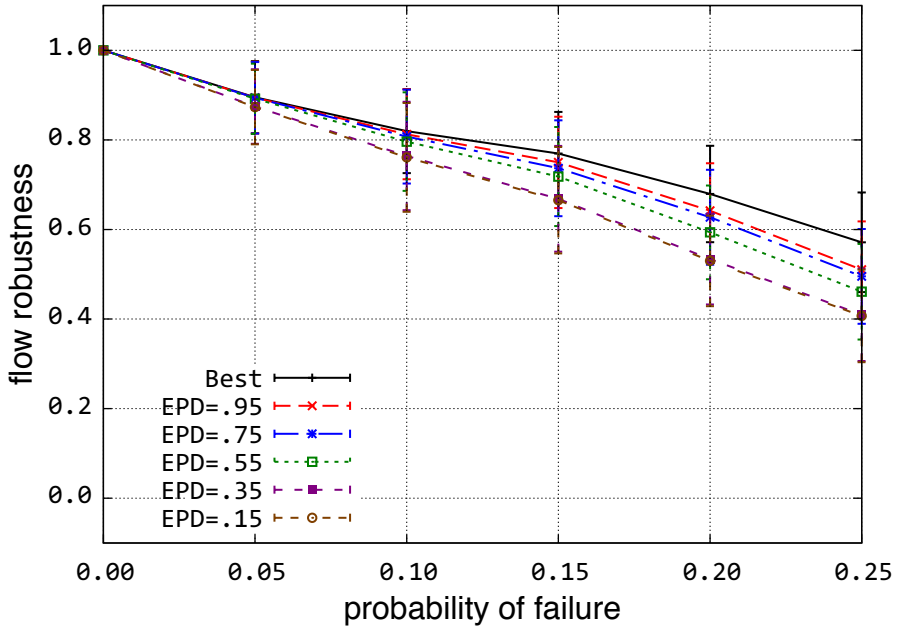

Figure E.78: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Exodus logical topology

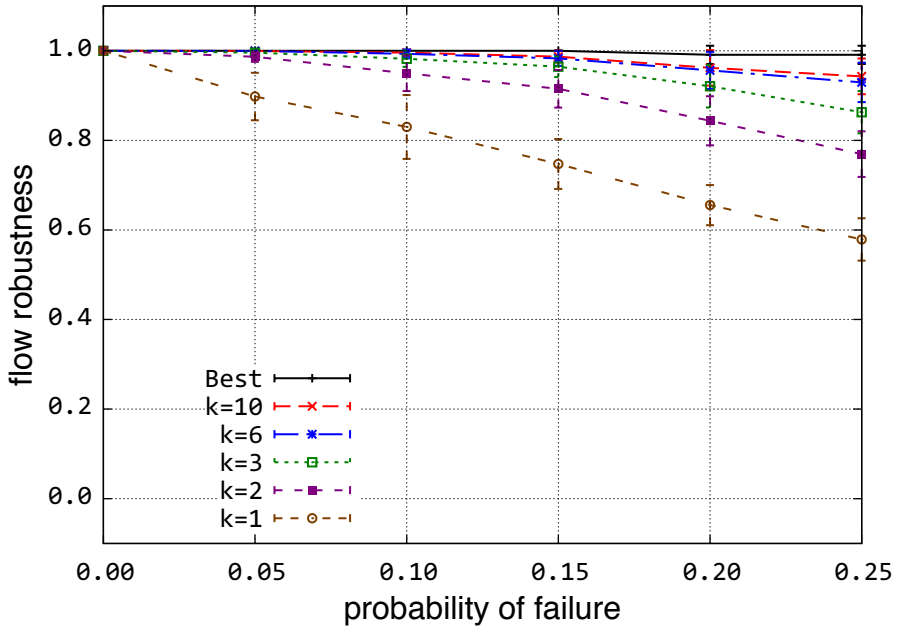

Figure E.79: Flow robustness vs. link failure probability for the Exodus logical topology

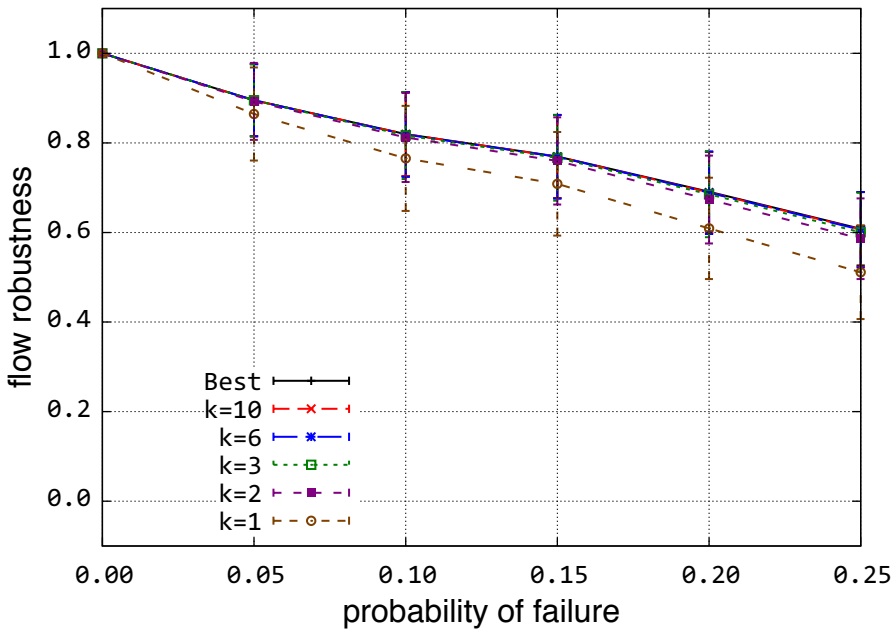

Figure E.80: Flow robustness vs. node failure probability for the Exodus logical topology

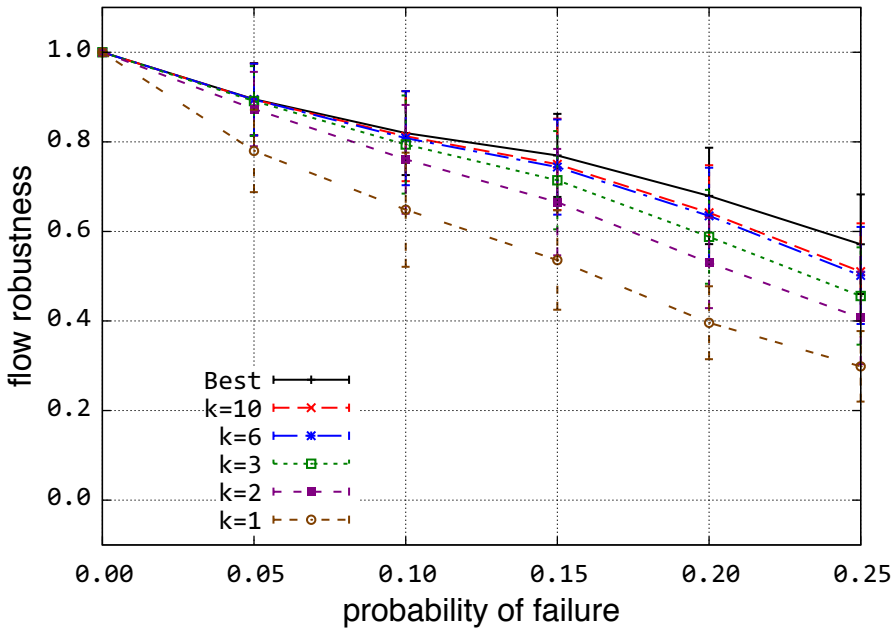

Figure E.81: Flow robustness vs. node & link failure probability for the Exodus logical topology

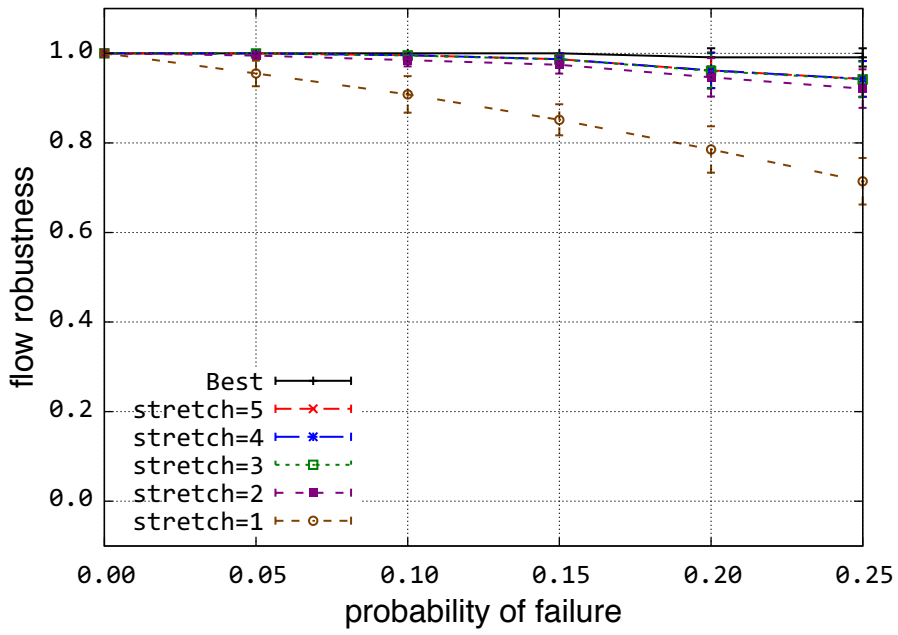

Figure E.82: Flow robustness vs. link failure probability for various stretch limits on the Exodus logical topology

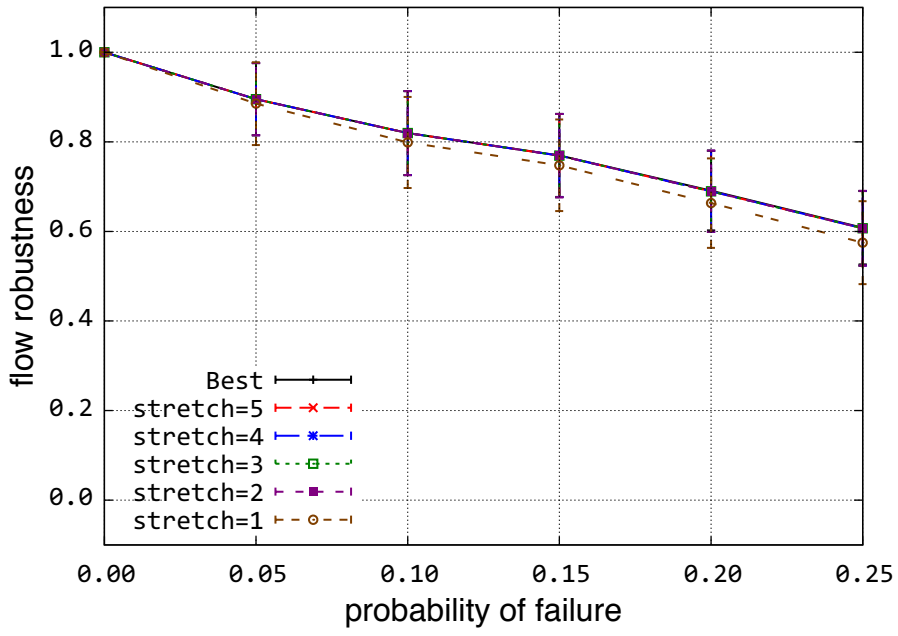

Figure E.83: Flow robustness vs. node failure probability for various stretch limits on the Exodus logical topology

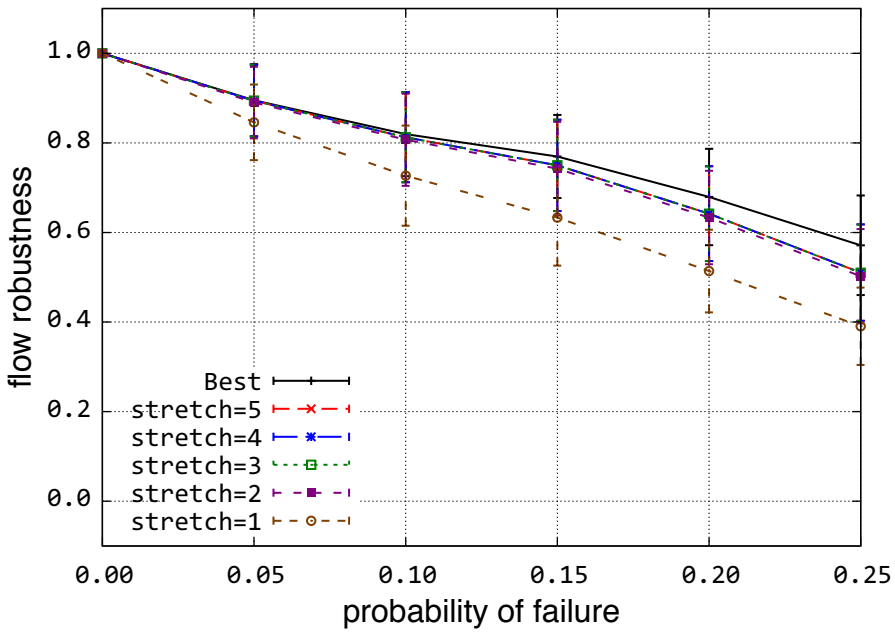

Figure E.84: Flow robustness vs. node & link failure probability for various stretch limits on the Exodus logical topology

E.2.5 Level-3 logical

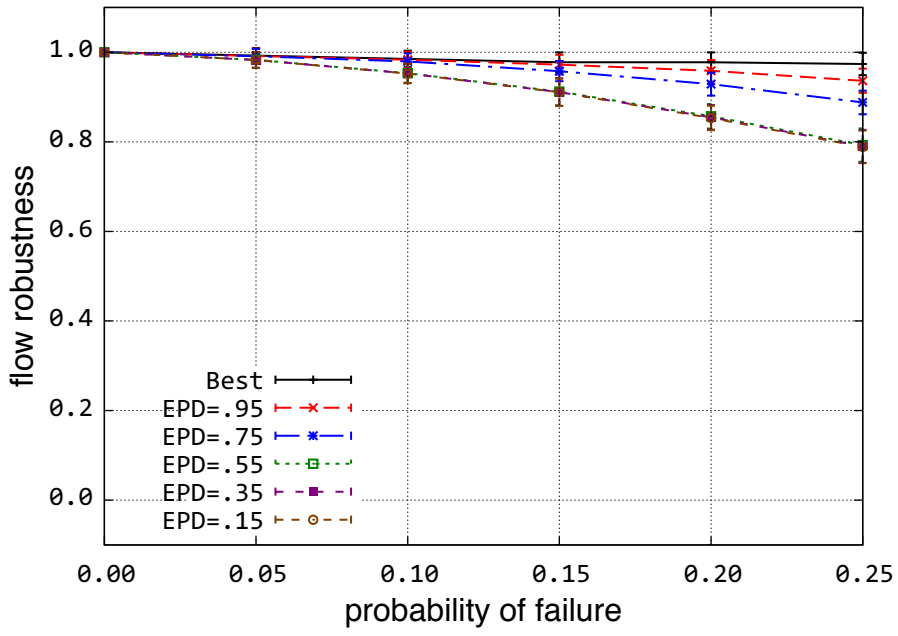

Figure E.85: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Level-3 logical topology

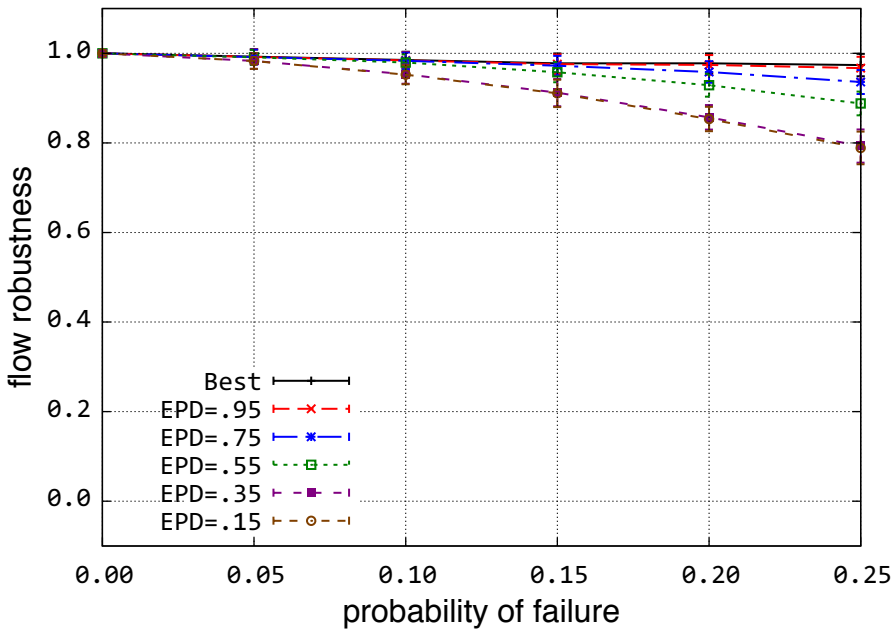

Figure E.86: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Level-3 logical topology

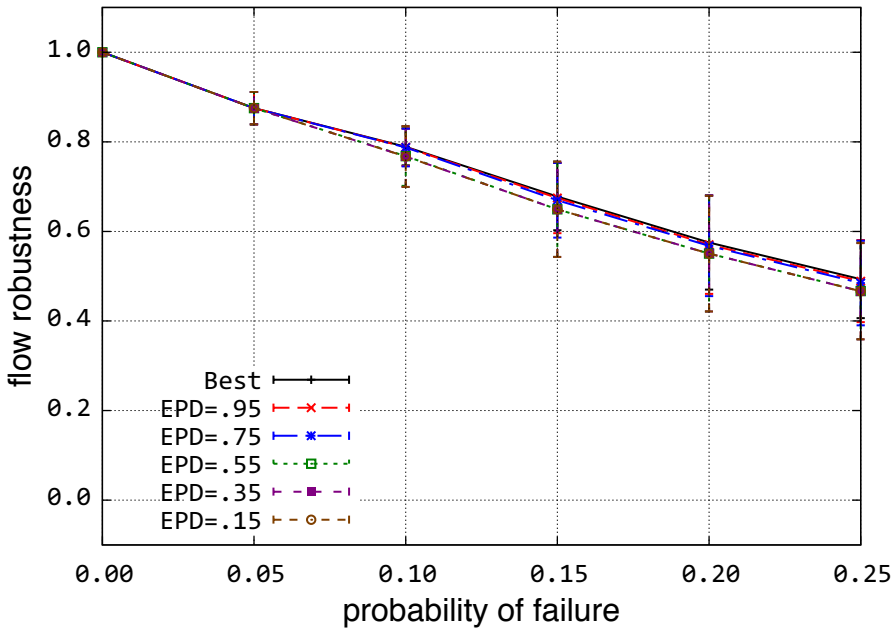

Figure E.87: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Level-3 logical topology

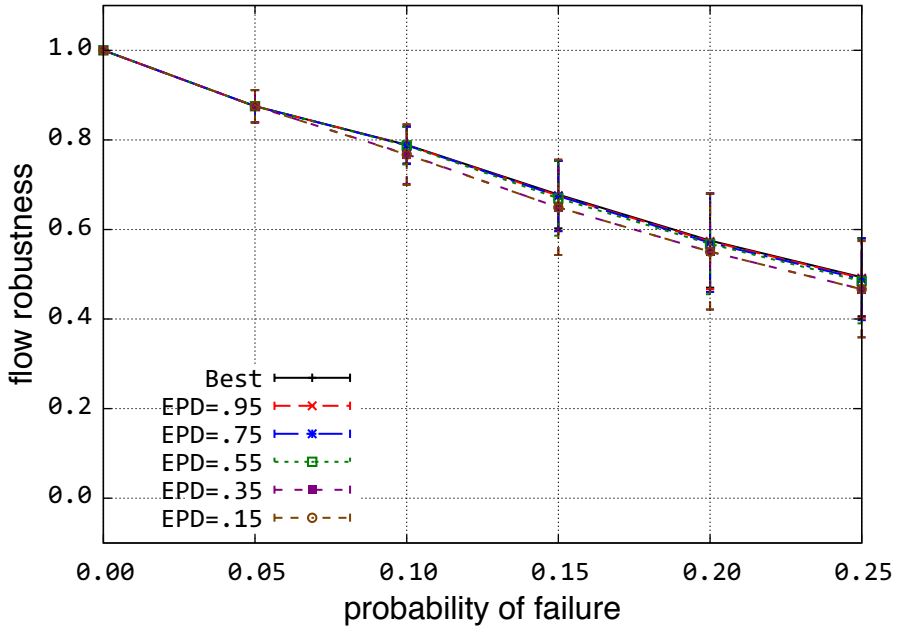

Figure E.88: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Level-3 logical topology

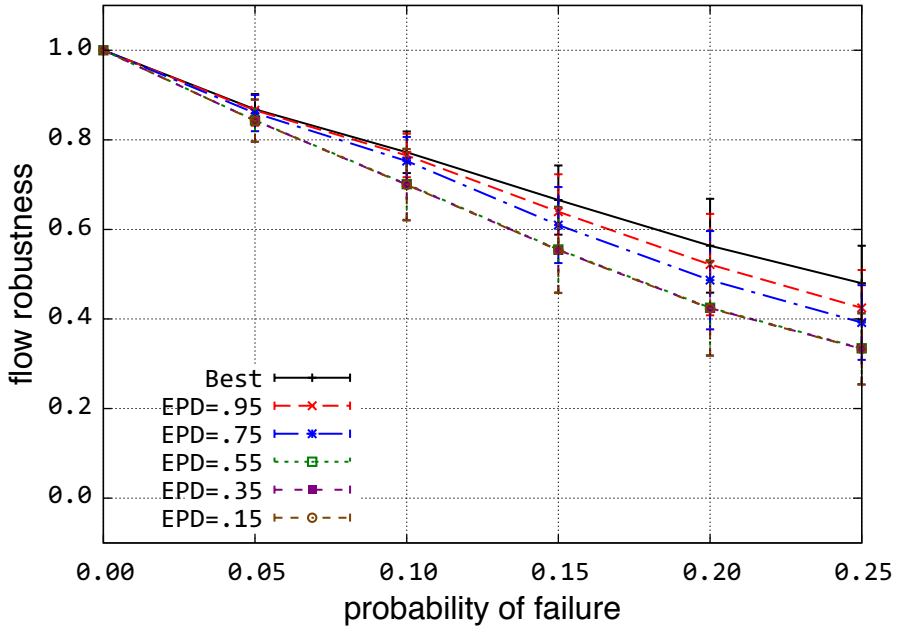

Figure E.89: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Level-3 logical topology

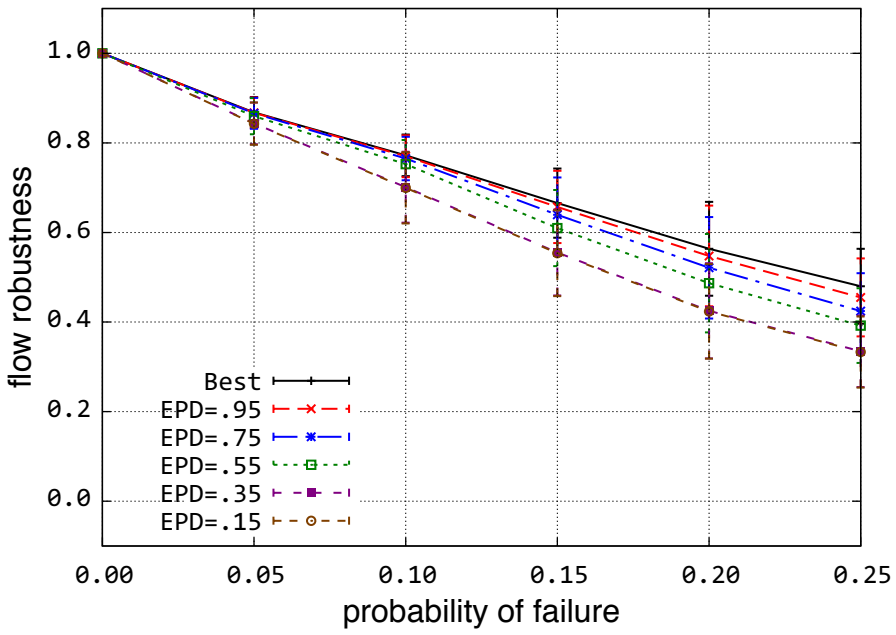

Figure E.90: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Level-3 logical topology

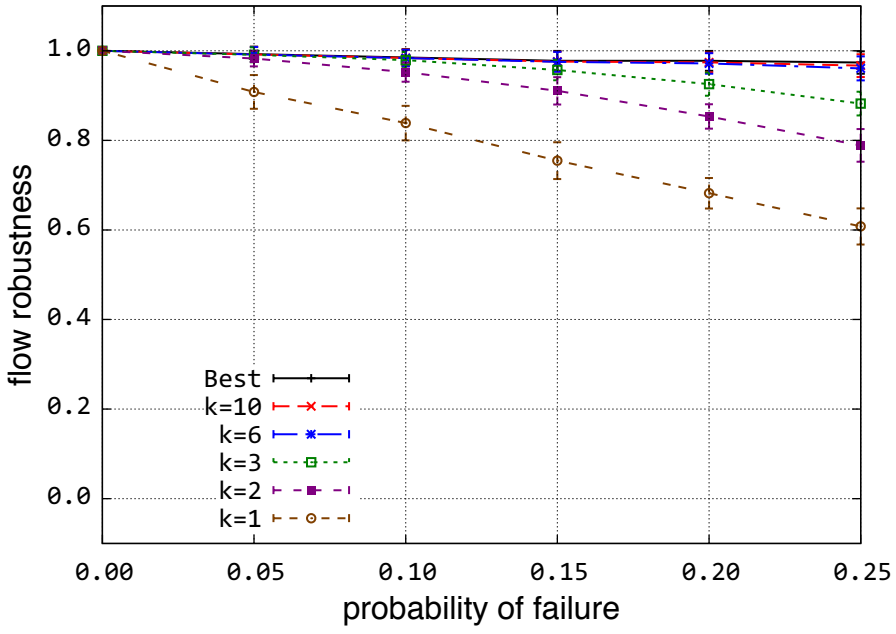

Figure E.91: Flow robustness vs. link failure probability for the Level-3 logical topology

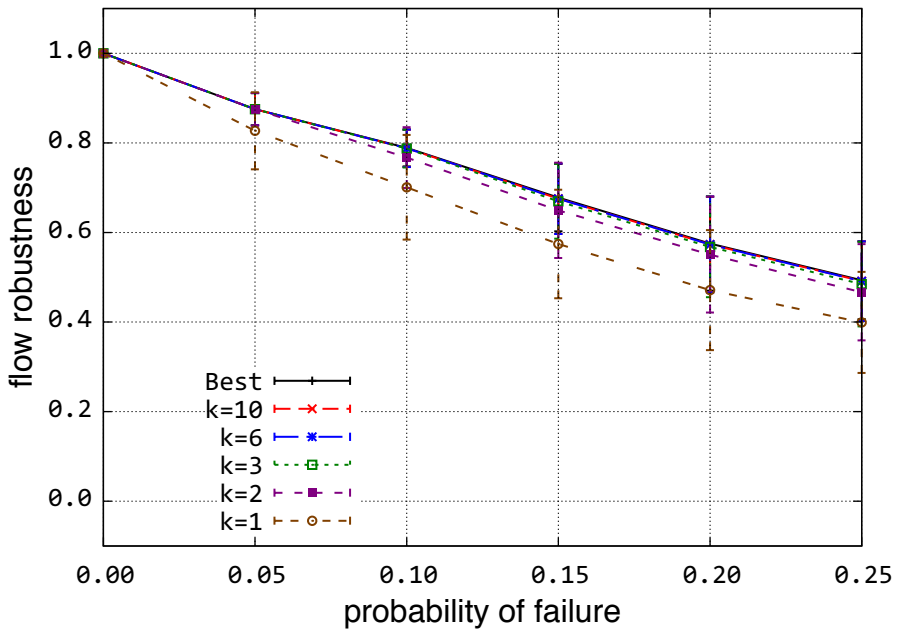

Figure E.92: Flow robustness vs. node failure probability for the Level-3 logical topology

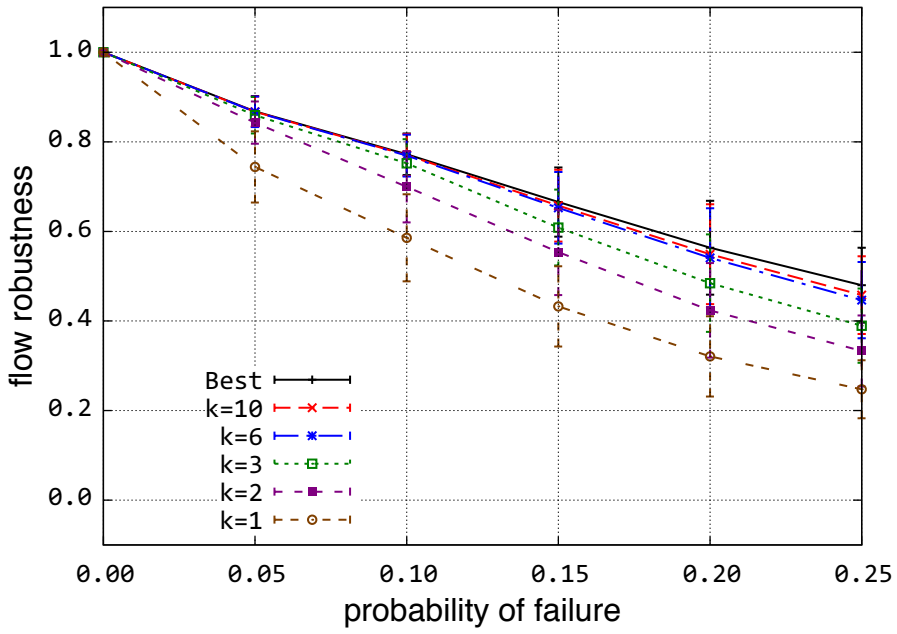

Figure E.93: Flow robustness vs. node & link failure probability for the Level-3 logical topology

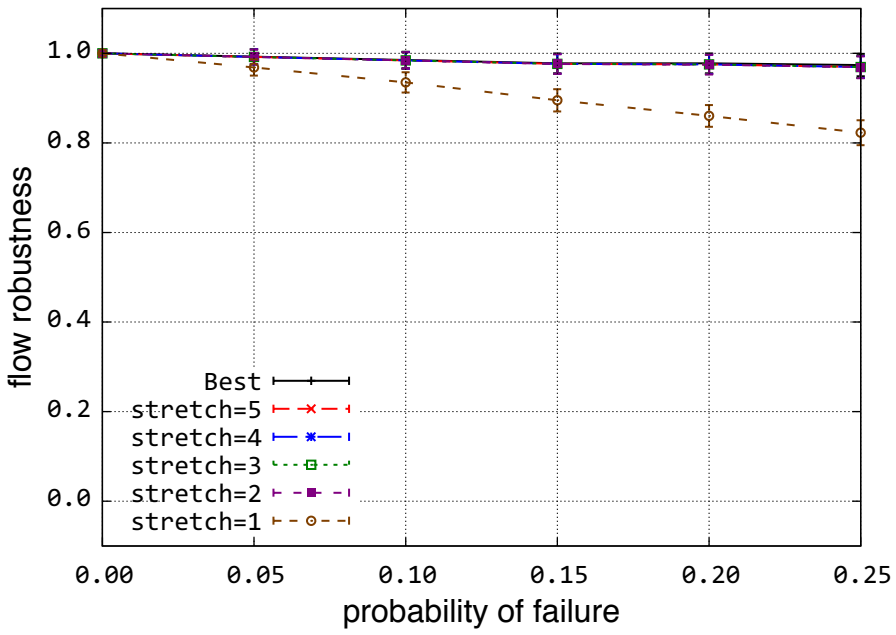

Figure E.94: Flow robustness vs. link failure probability for various stretch limits on the Level-3 logical topology

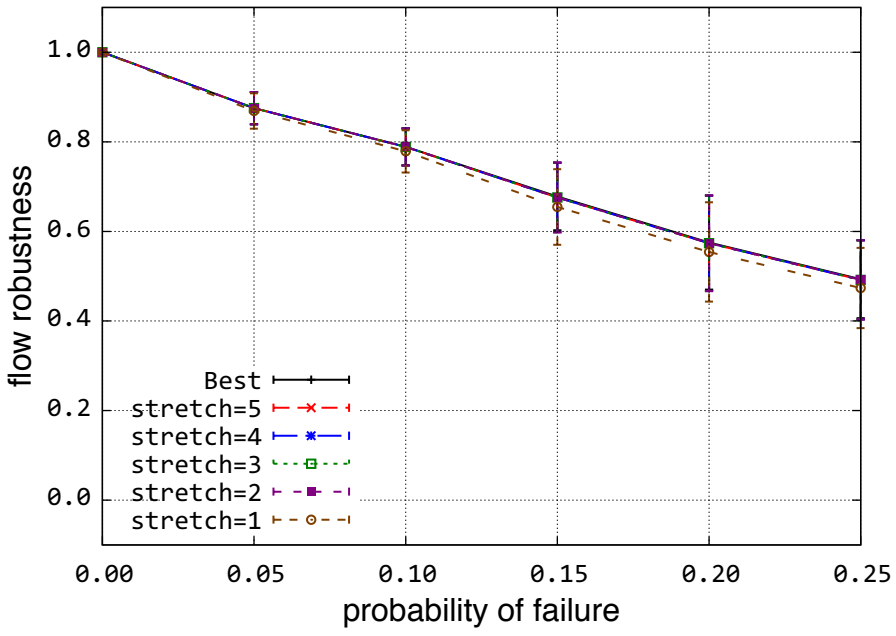

Figure E.95: Flow robustness vs. node failure probability for various stretch limits on the Level-3 logical topology

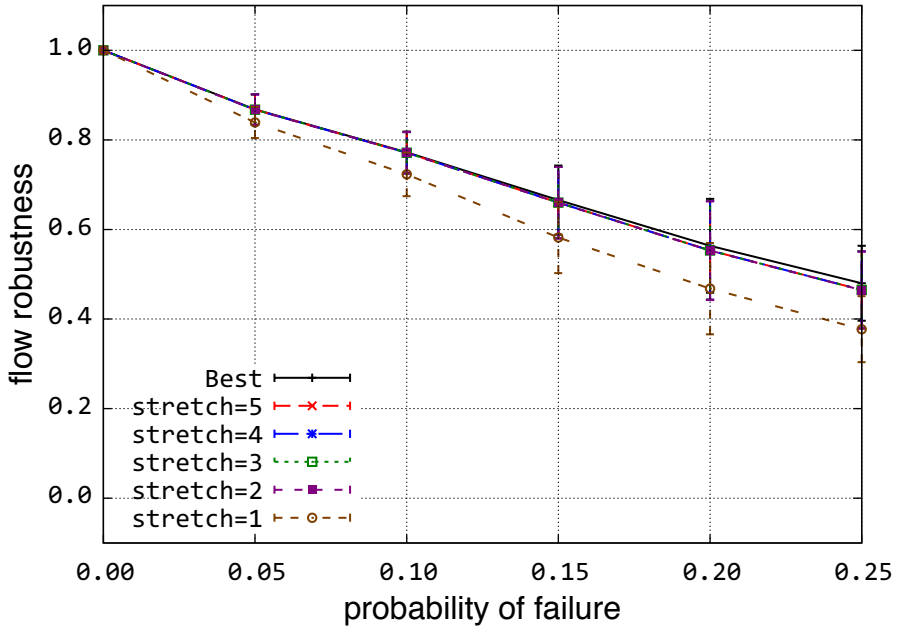

Figure E.96: Flow robustness vs. node & link failure probability for various stretch limits on the Level-3 logical topology

E.2.6 Sprint logical

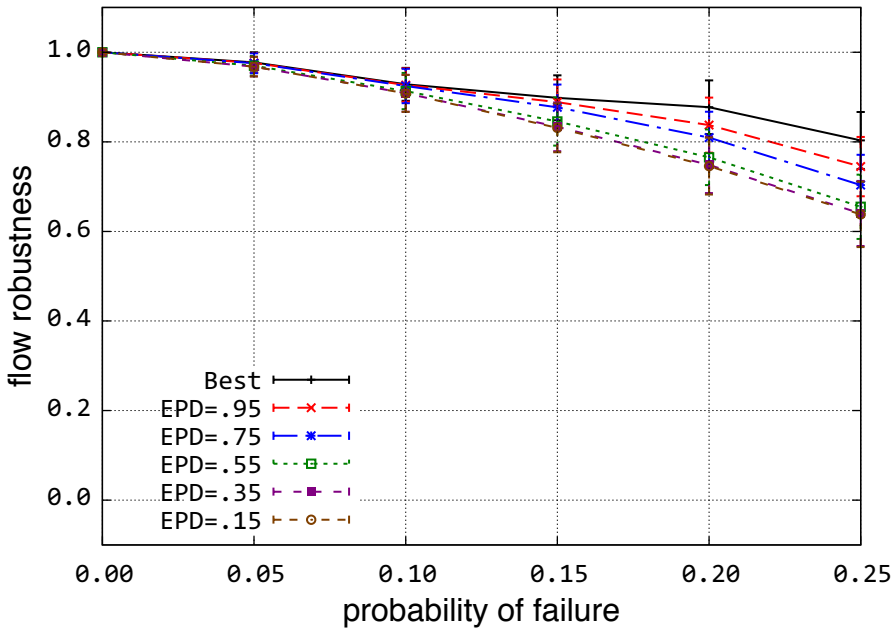

Figure E.97: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Sprint logical topology

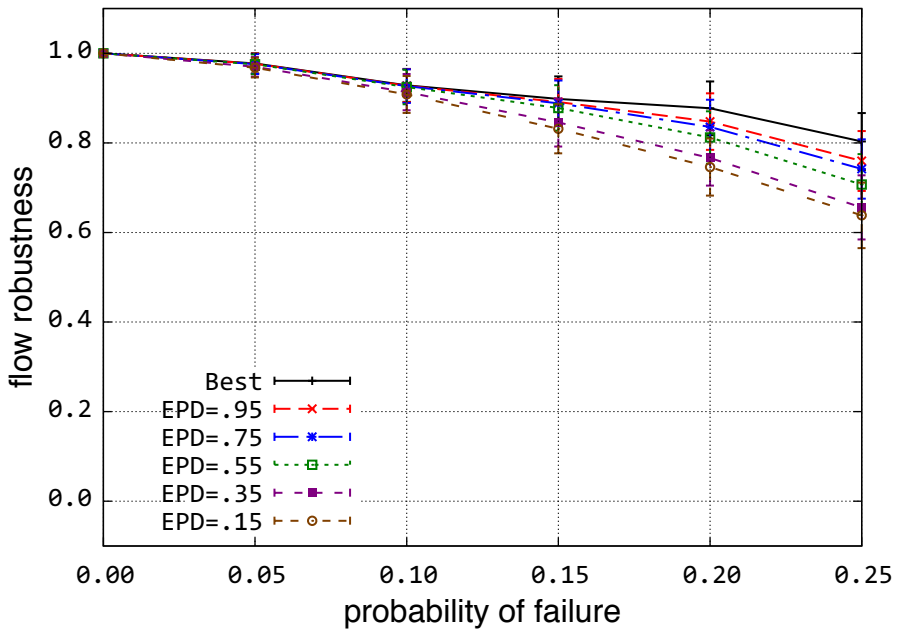

Figure E.98: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Sprint logical topology

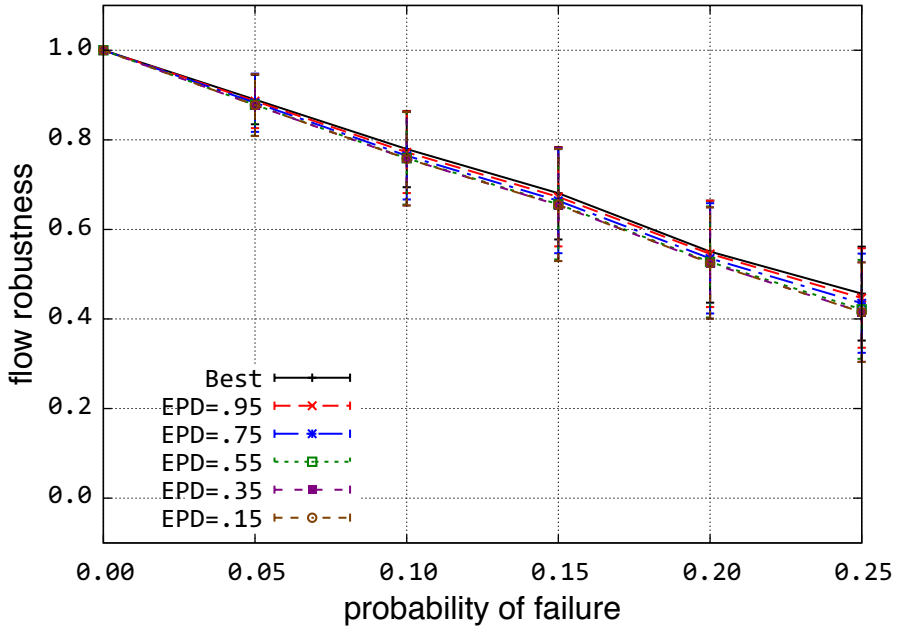

Figure E.99: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Sprint logical topology

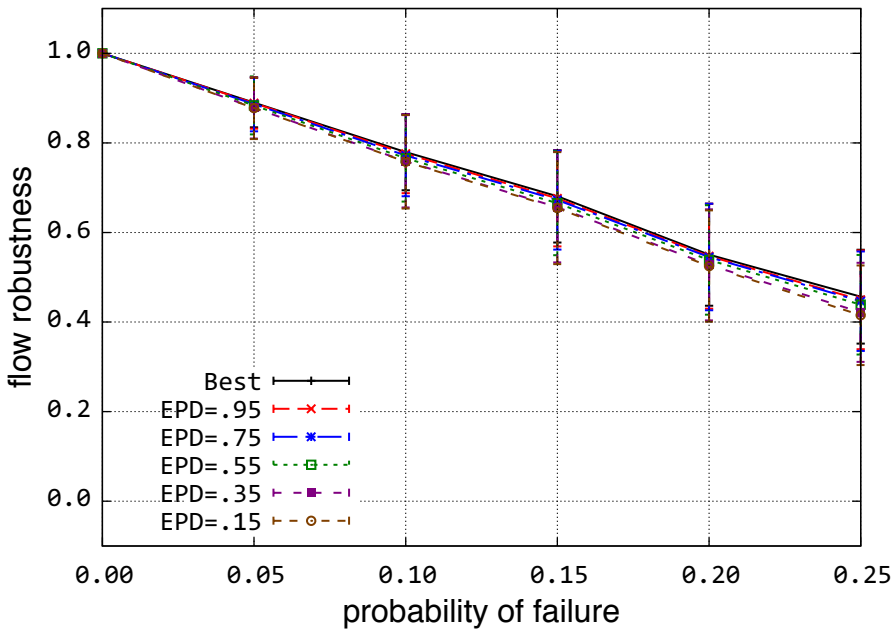

Figure E.100: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Sprint logical topology

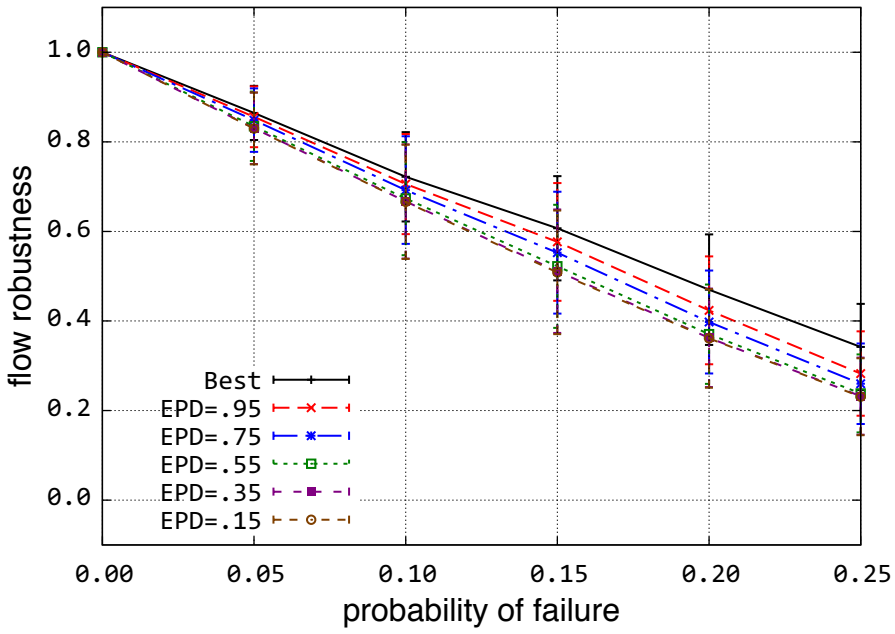

Figure E.101: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Sprint logical topology

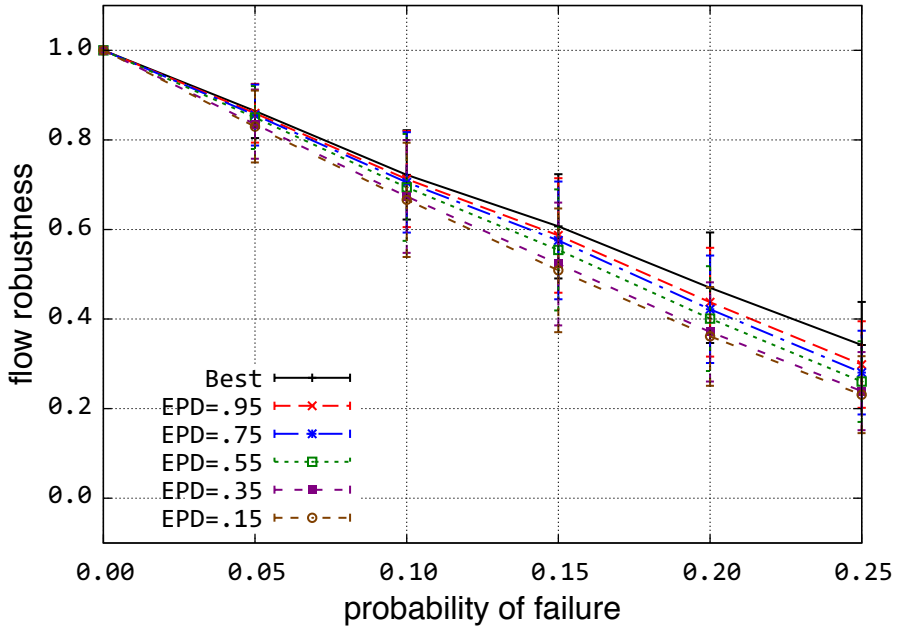

Figure E.102: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Sprint logical topology

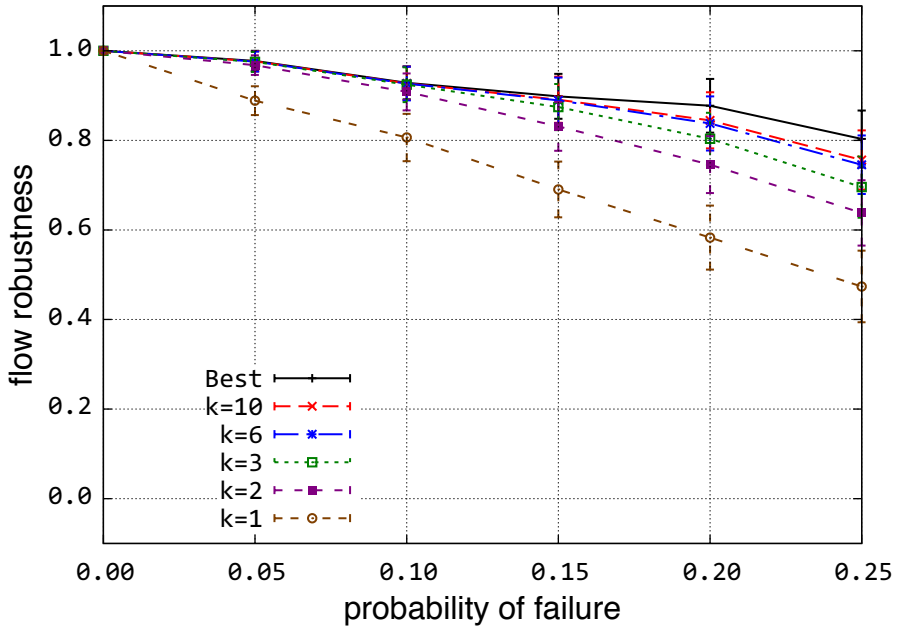

Figure E.103: Flow robustness vs. link failure probability for the Sprint logical topology

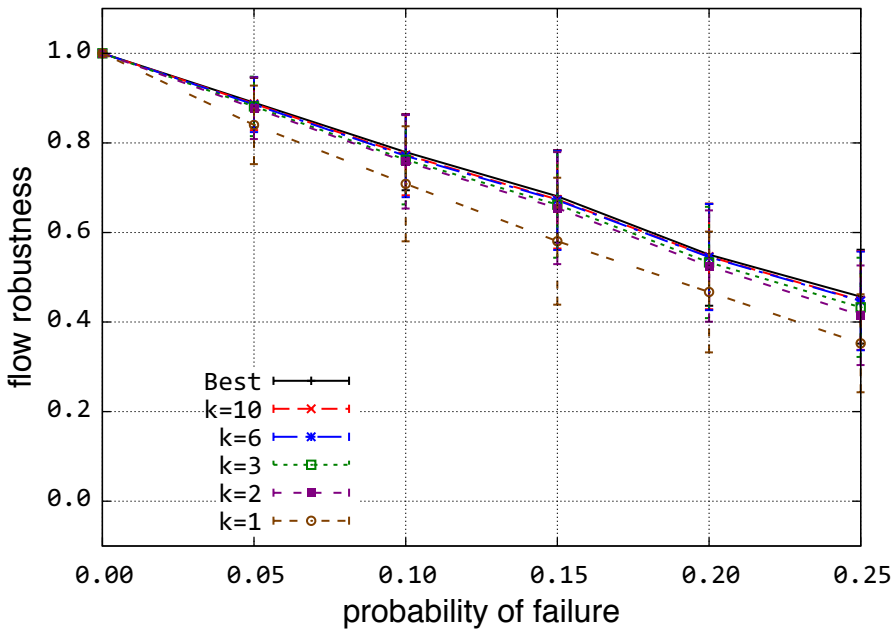

Figure E.104: Flow robustness vs. node failure probability for the Sprint logical topology

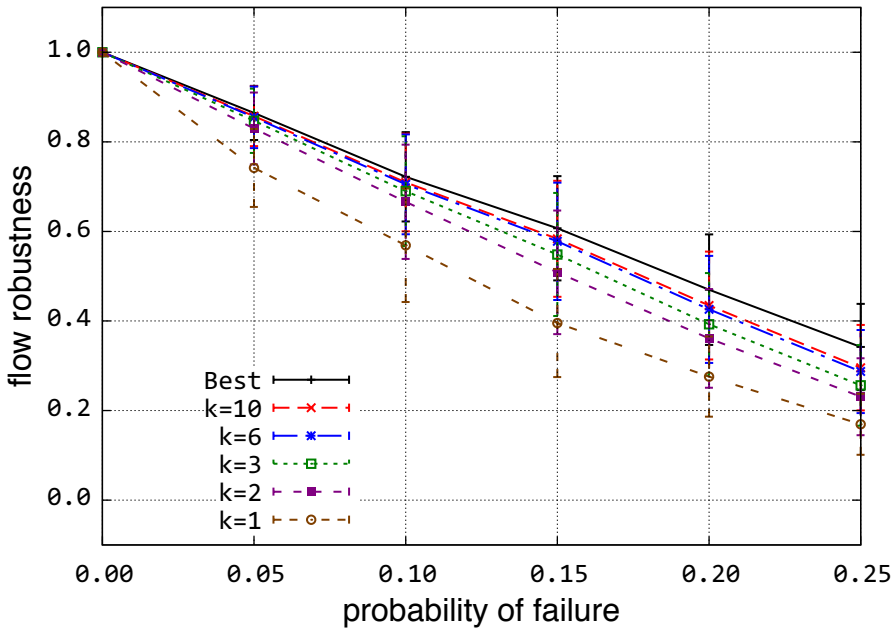

Figure E.105: Flow robustness vs. node & link failure probability for the Sprint logical topology

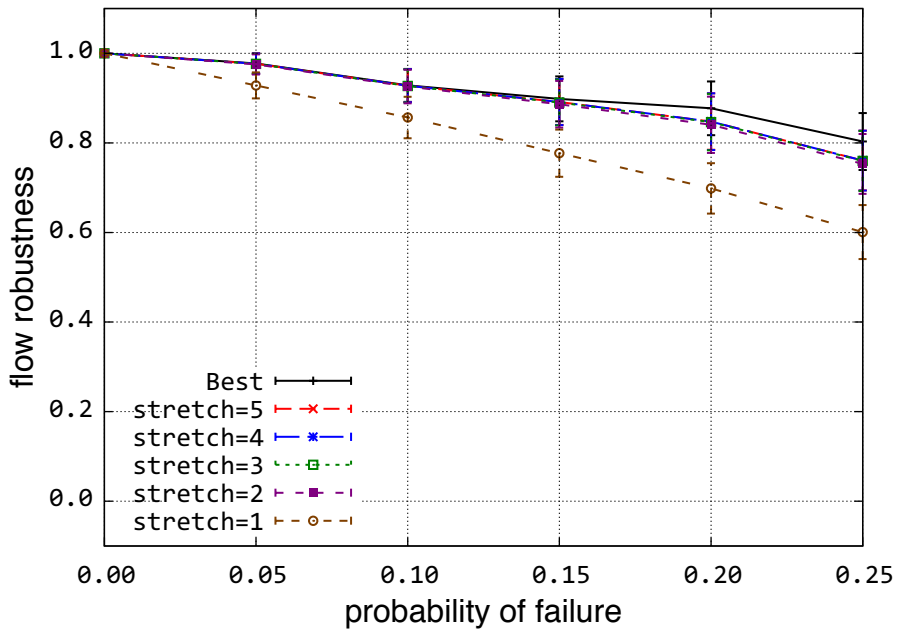

Figure E.106: Flow robustness vs. link failure probability for various stretch limits on the Sprint logical topology

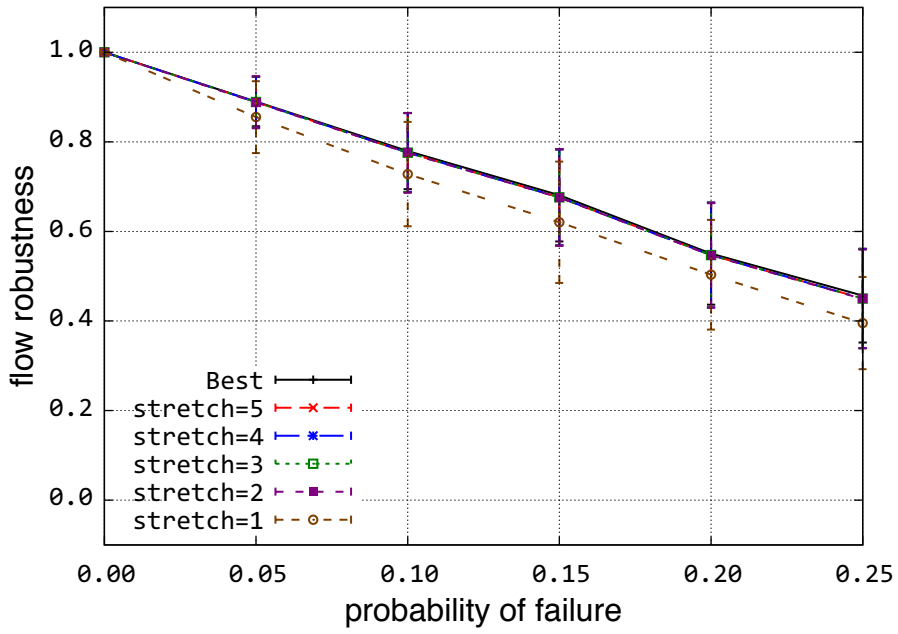

Figure E.107: Flow robustness vs. node failure probability for various stretch limits on the Sprint logical topology

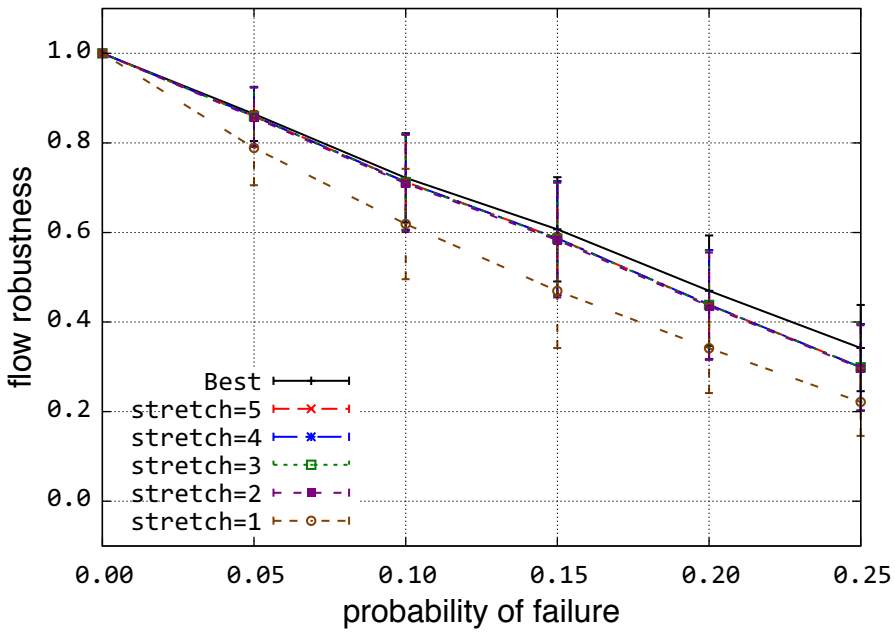

Figure E.108: Flow robustness vs. node & link failure probability for various stretch limits on the Sprint logical topology

E.2.7 Telstra logical

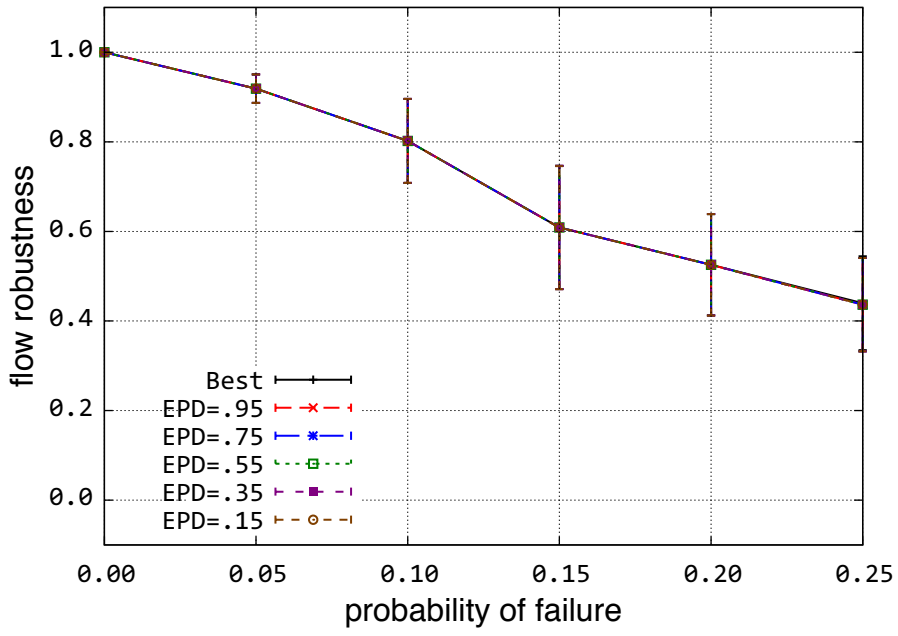

Figure E.109: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Telstra logical topology

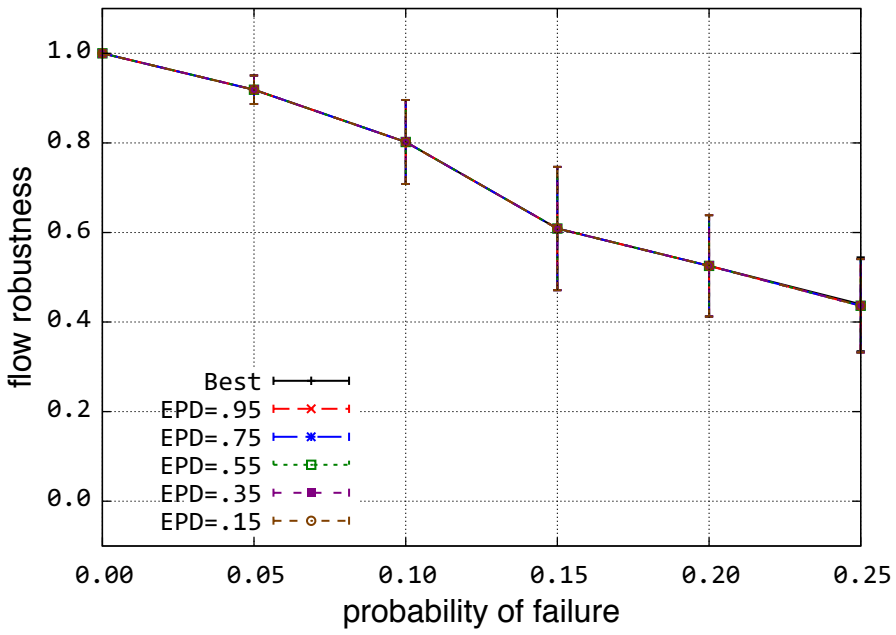

Figure E.110: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Telstra logical topology

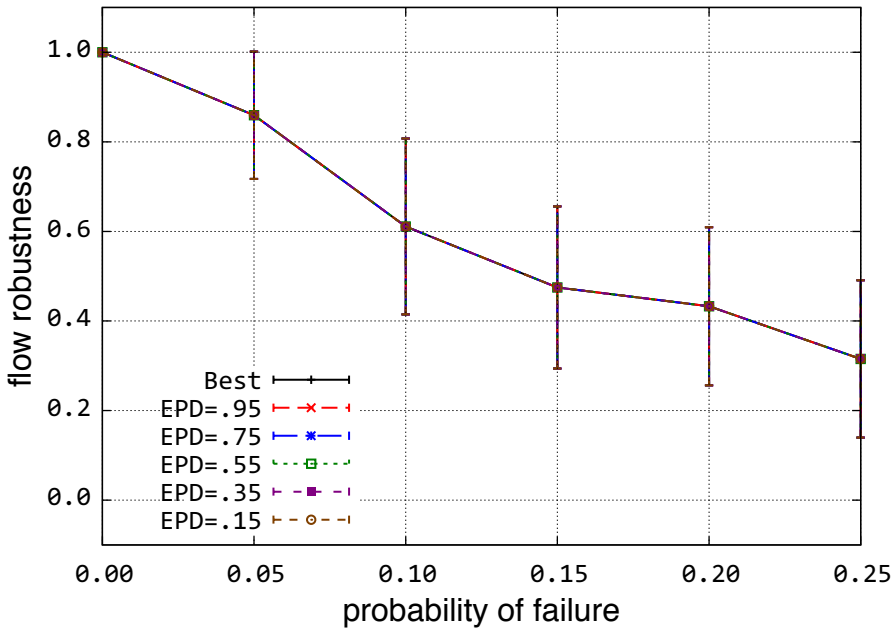

Figure E.111: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Telstra logical topology

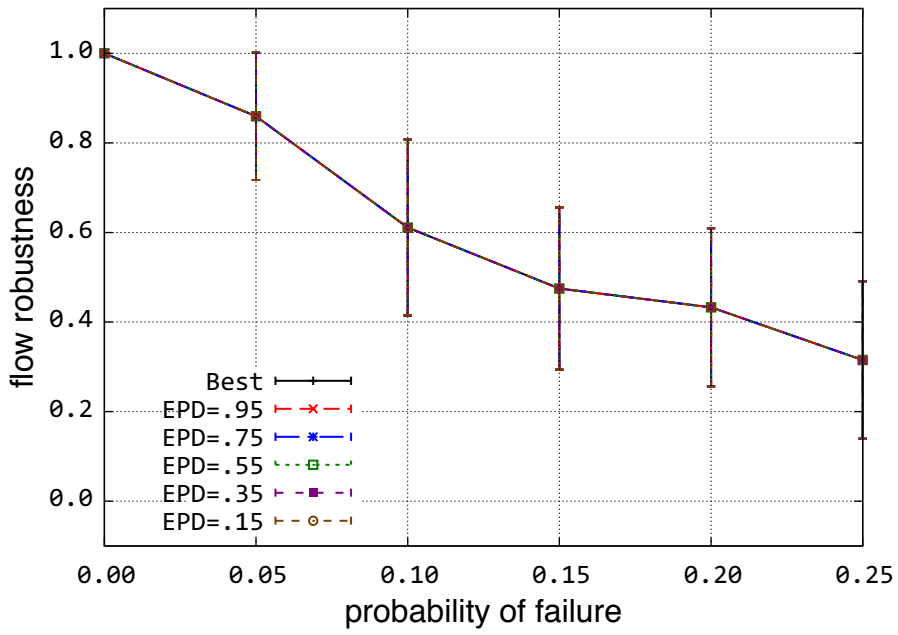

Figure E.112: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Telstra logical topology

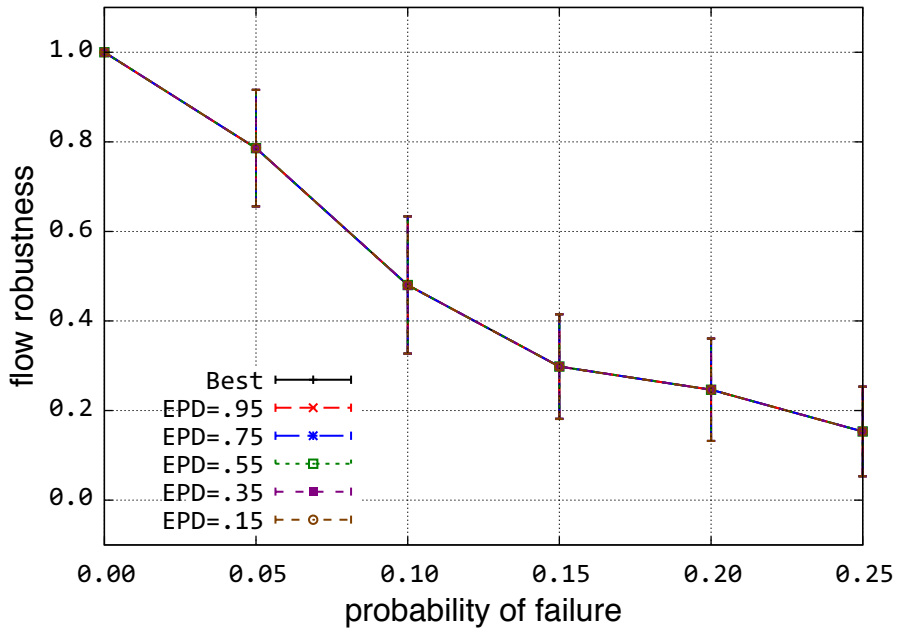

Figure E.113: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Telstra logical topology

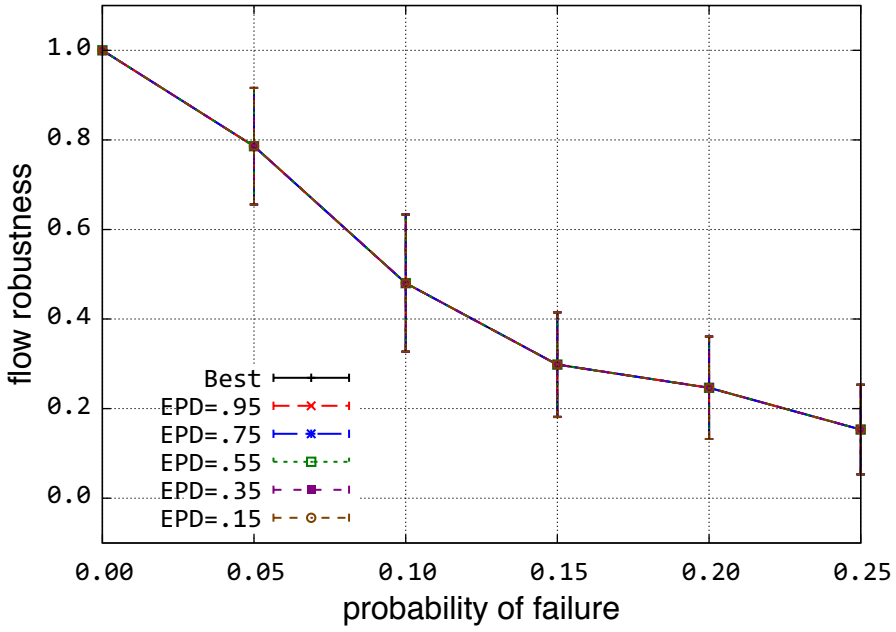

Figure E.114: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Telstra logical topology

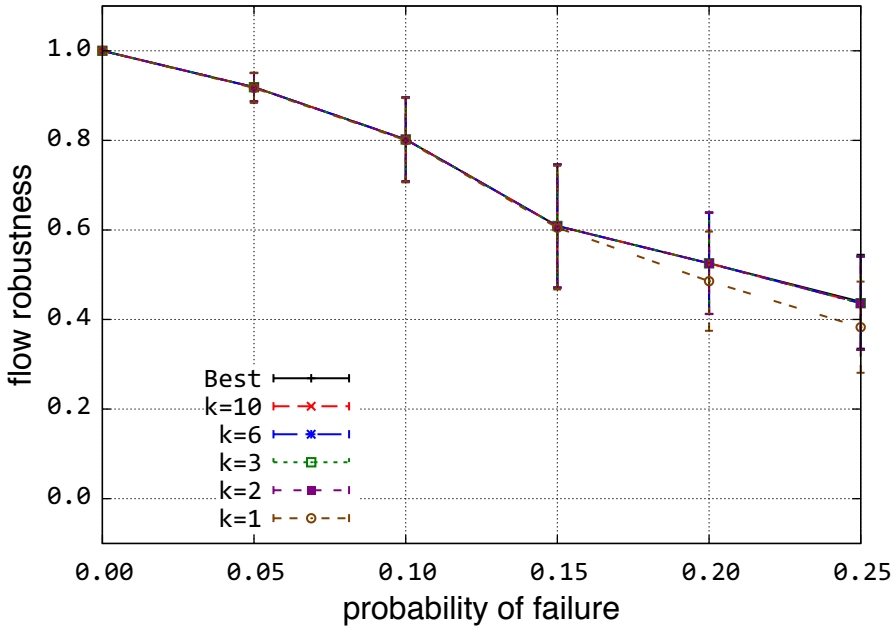

Figure E.115: Flow robustness vs. link failure probability for the Telstra logical topology

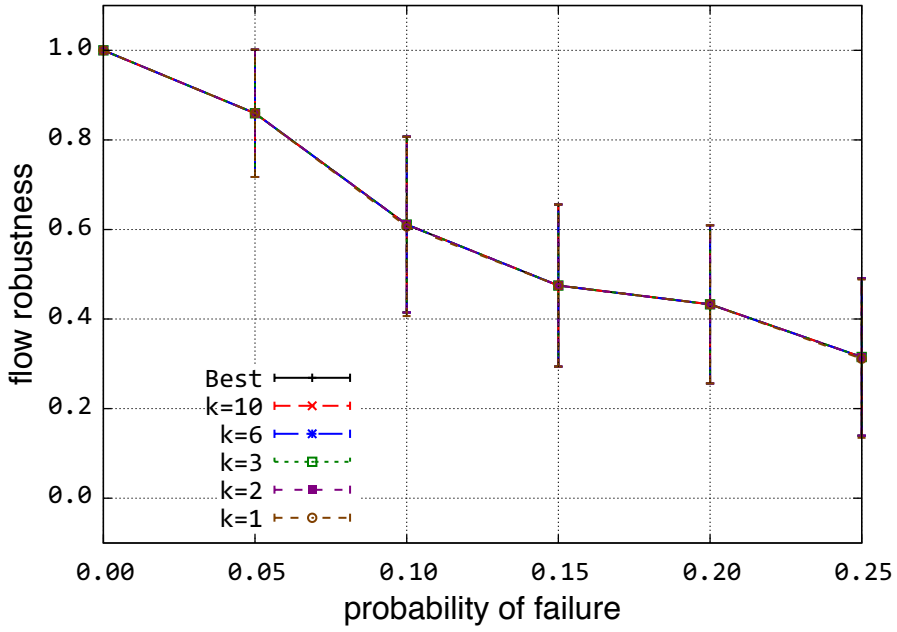

Figure E.116: Flow robustness vs. node failure probability for the Telstra logical topology

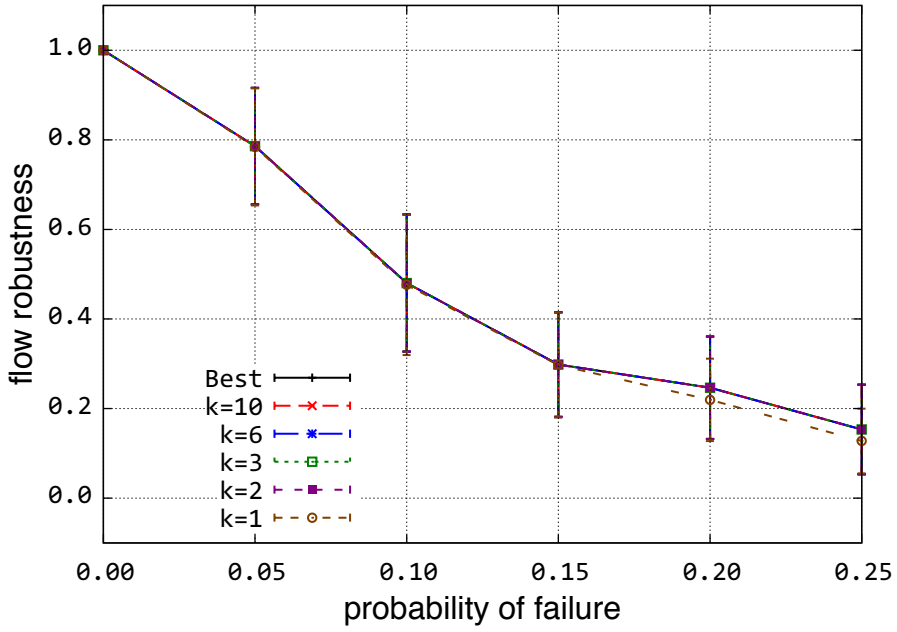

Figure E.117: Flow robustness vs. node & link failure probability for the Telstra logical topology

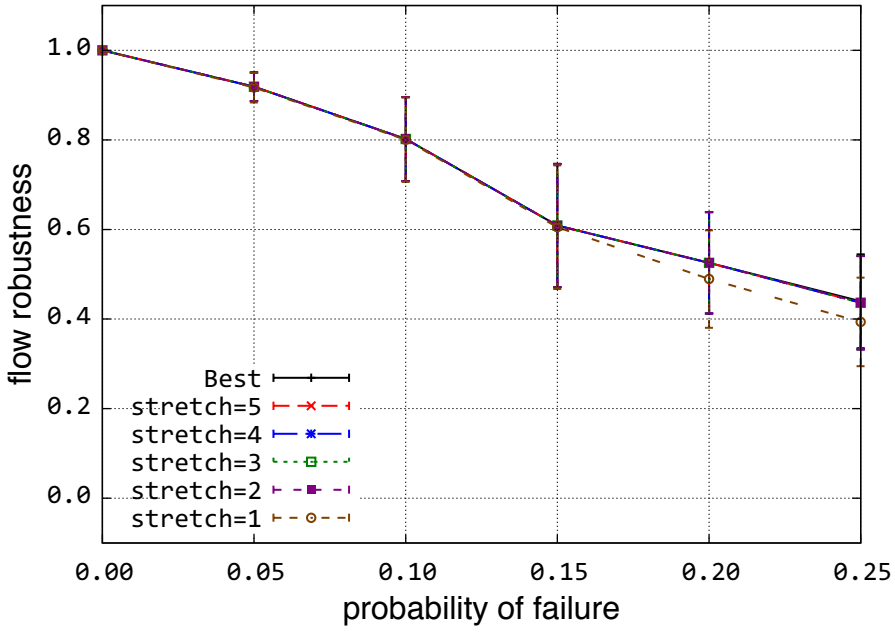

Figure E.118: Flow robustness vs. link failure probability for various stretch limits on the Telstra logical topology

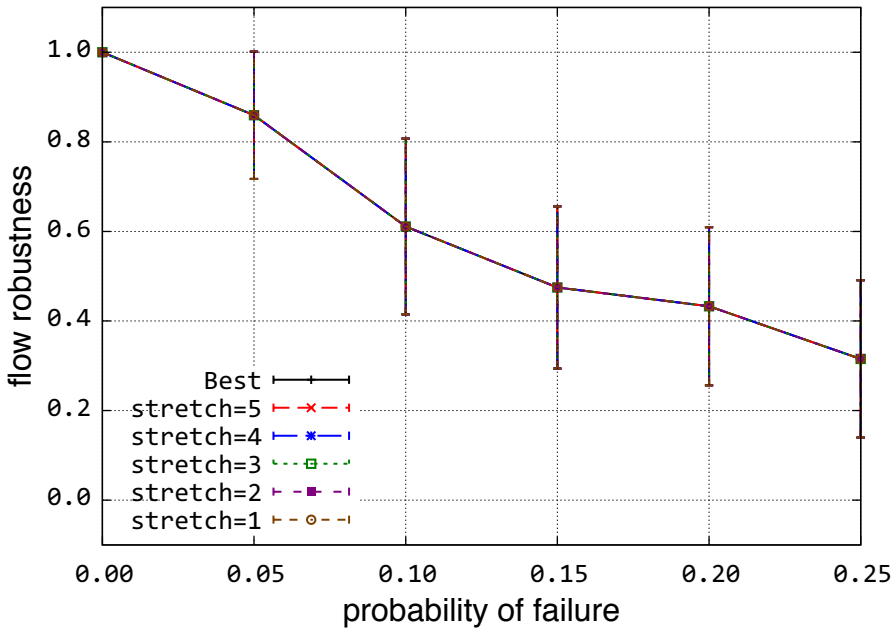

Figure E.119: Flow robustness vs. node failure probability for various stretch limits on the Telstra logical topology

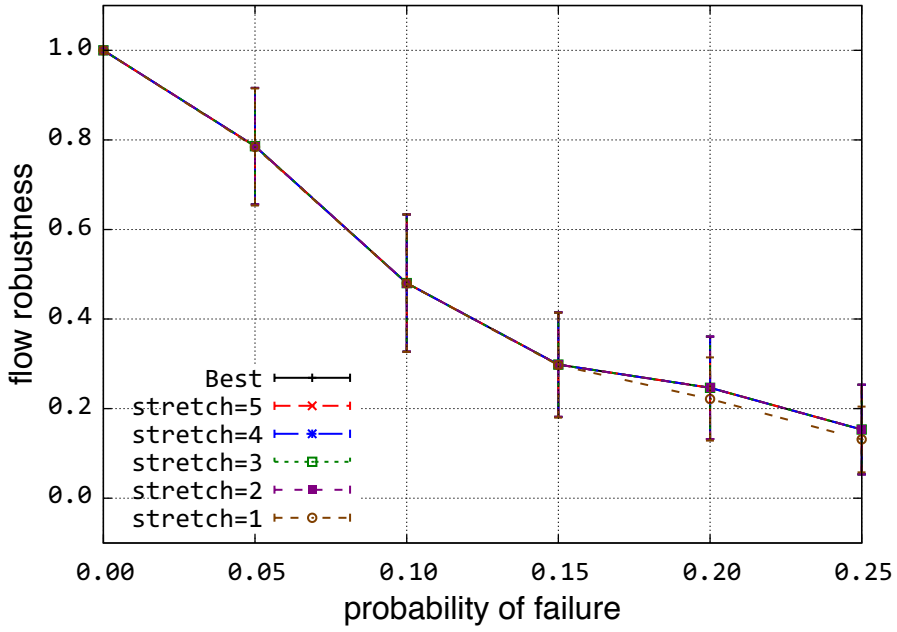

Figure E.120: Flow robustness vs. node & link failure probability for various stretch limits on the Telstra logical topology

E.2.8 Tiscali logical

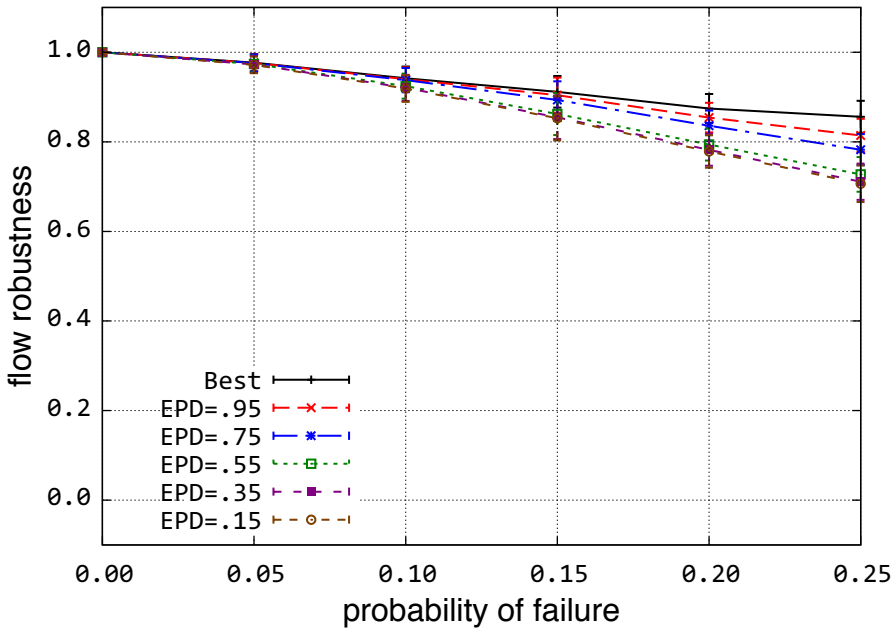

Figure E.121: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Tiscali logical topology

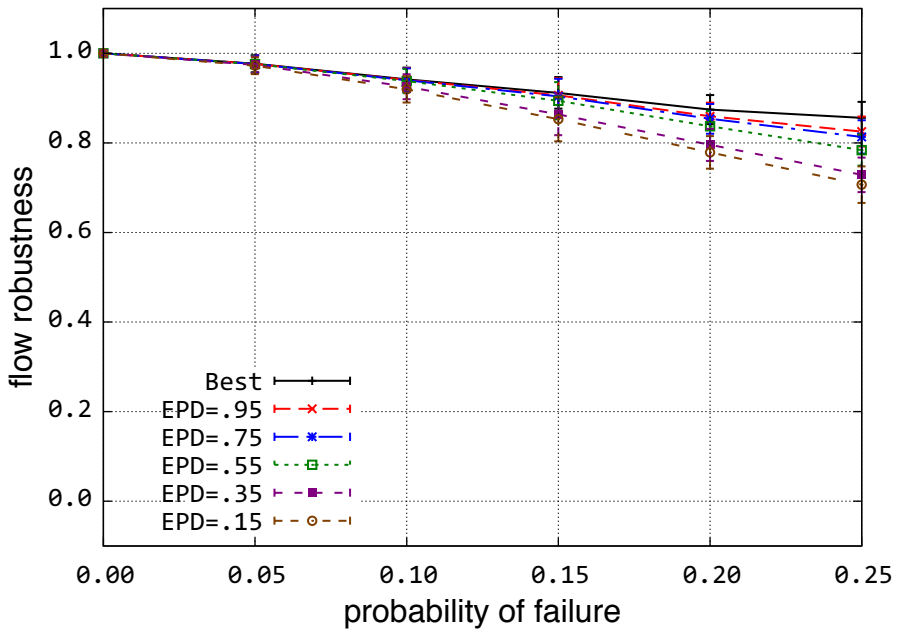

Figure E.122: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Tiscali logical topology

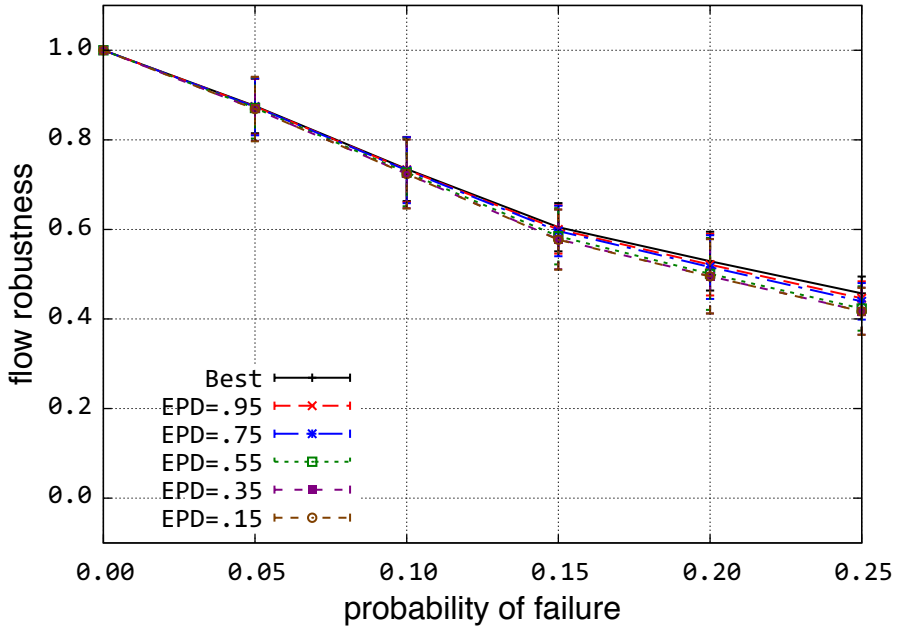

Figure E.123: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Tiscali logical topology

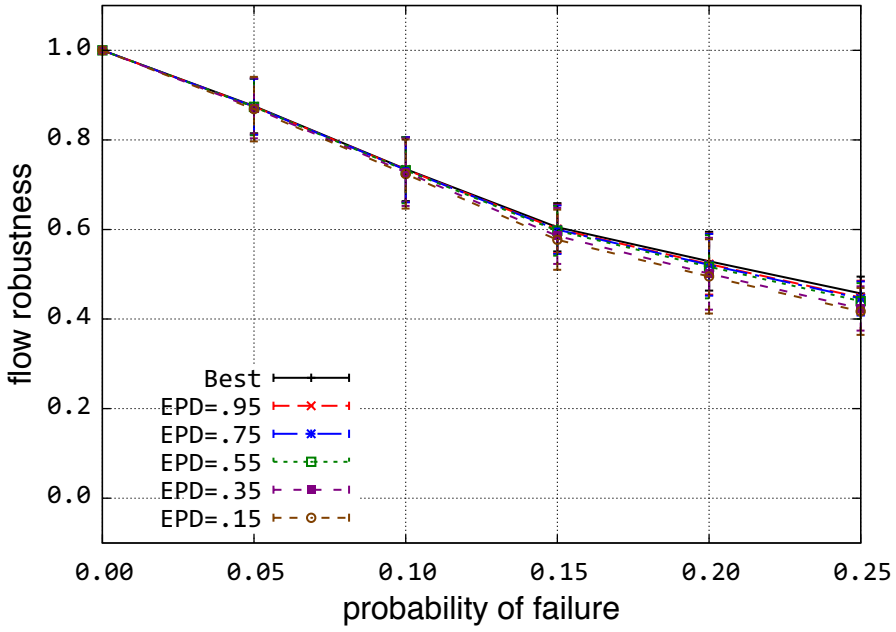

Figure E.124: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Tiscali logical topology

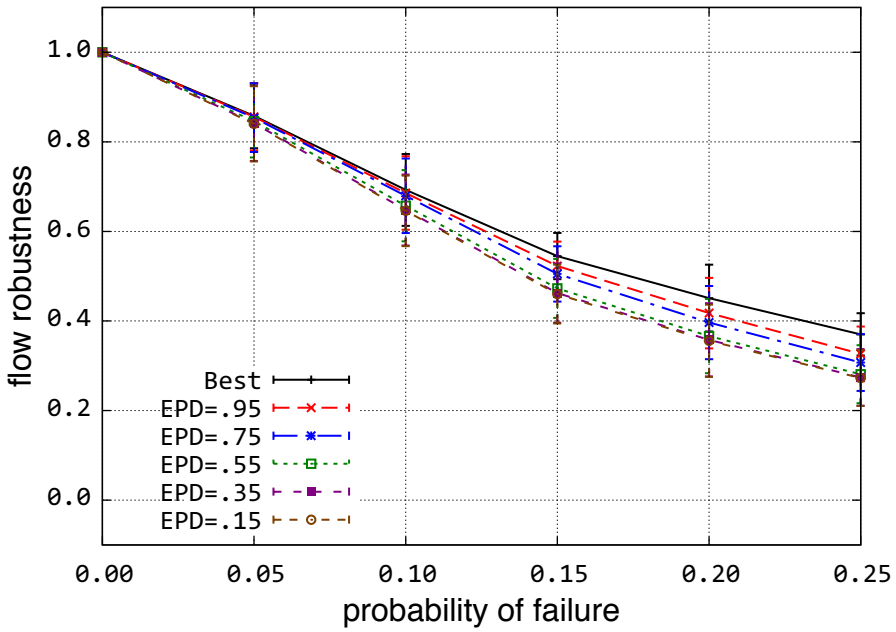

Figure E.125: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Tiscali logical topology

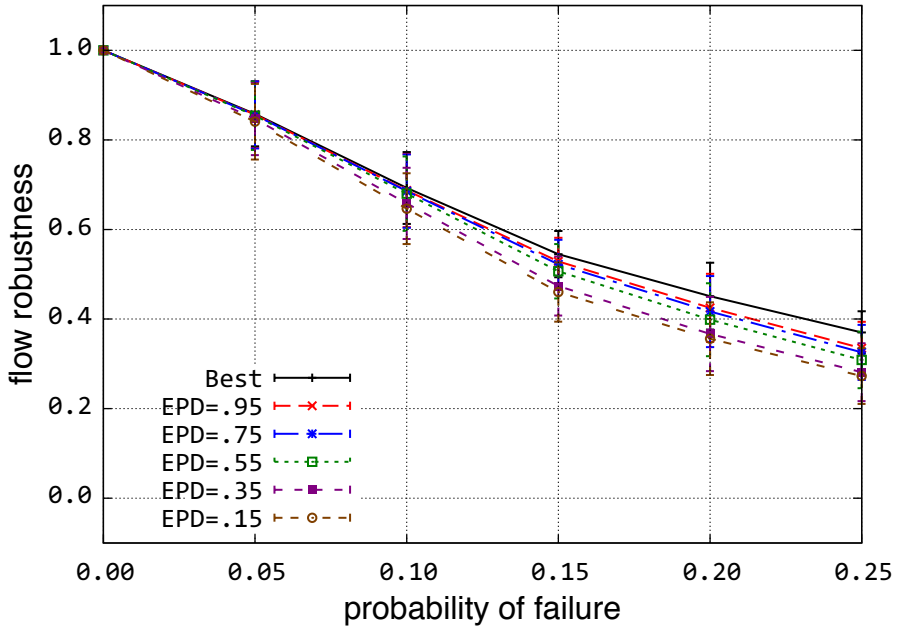

Figure E.126: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Tiscali logical topology

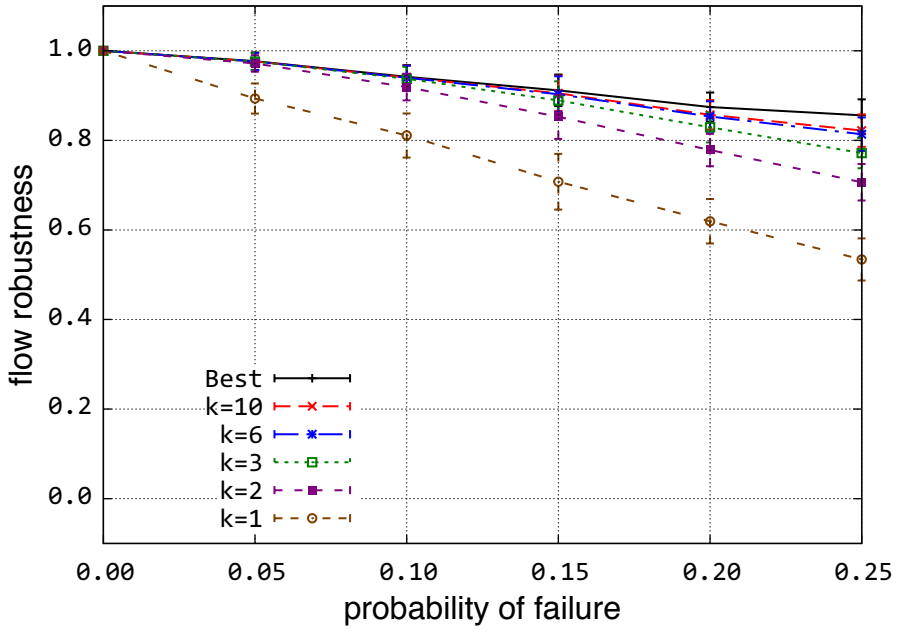

Figure E.127: Flow robustness vs. link failure probability for the Tiscali logical topology

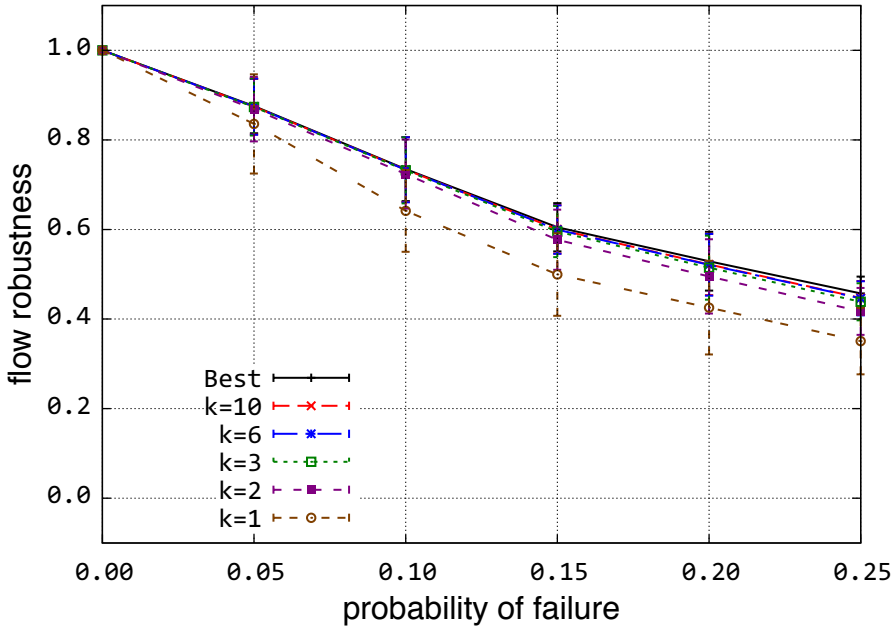

Figure E.128: Flow robustness vs. node failure probability for the Tiscali logical topology

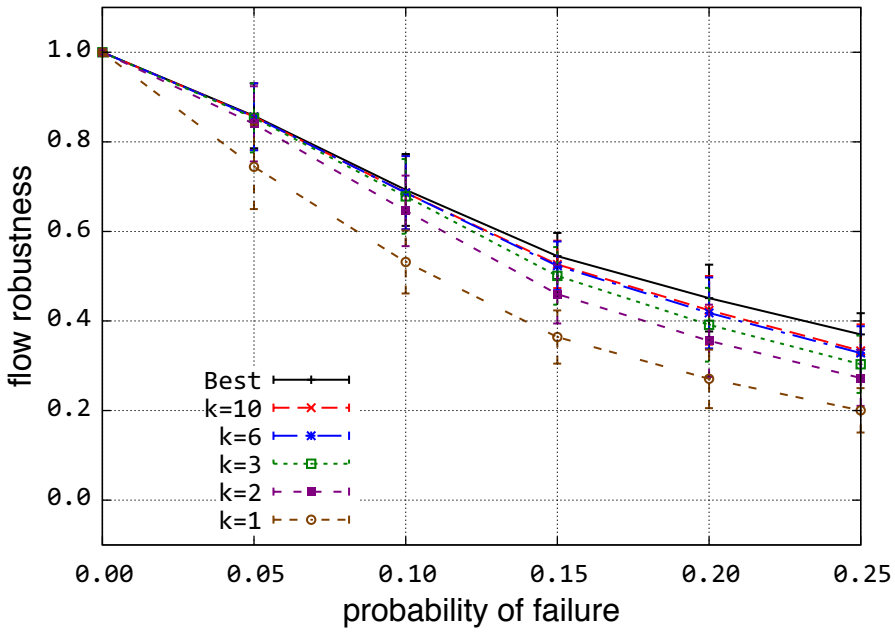

Figure E.129: Flow robustness vs. node & link failure probability for the Tiscali logical topology

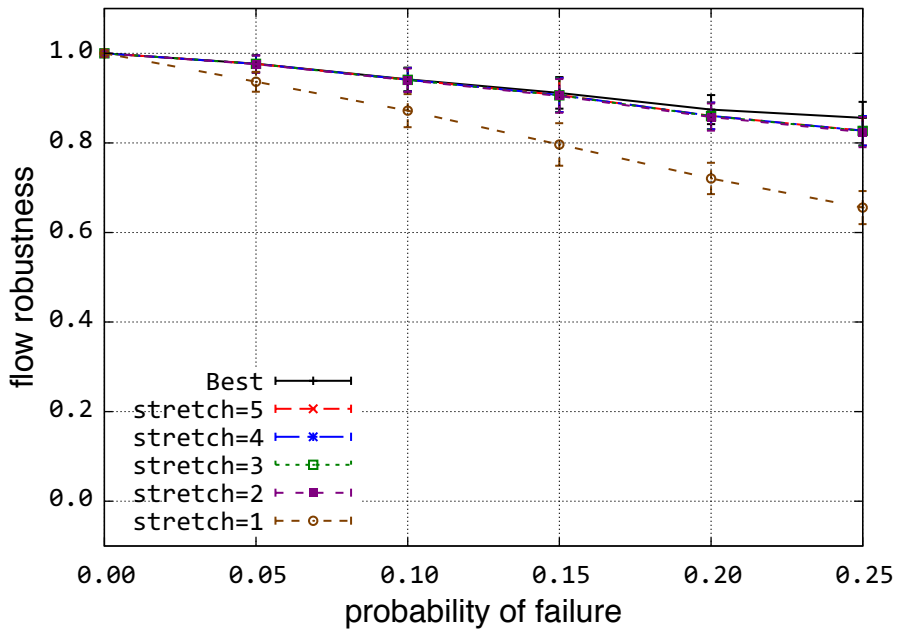

Figure E.130: Flow robustness vs. link failure probability for various stretch limits on the Tiscali logical topology

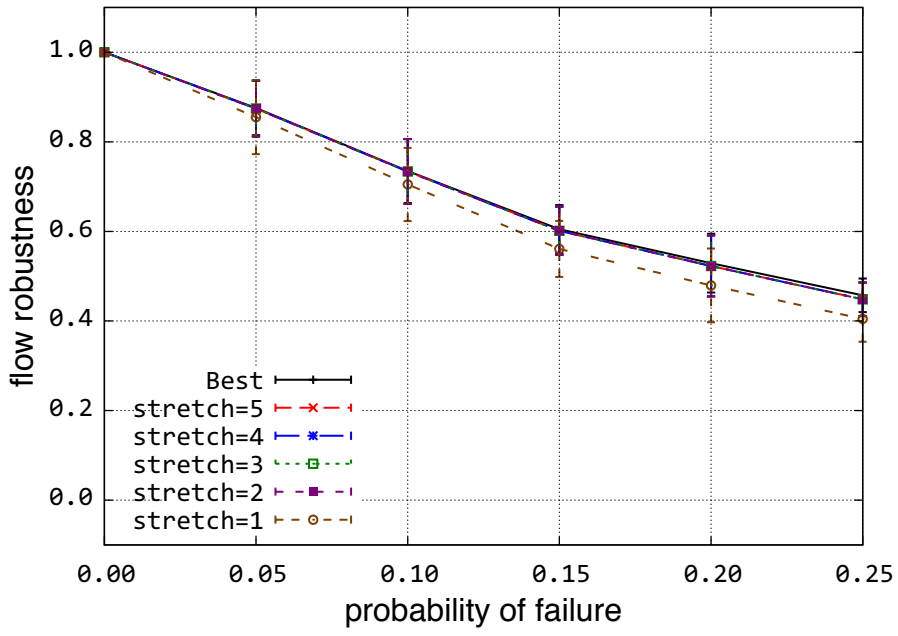

Figure E.131: Flow robustness vs. node failure probability for various stretch limits on the Tiscali logical topology

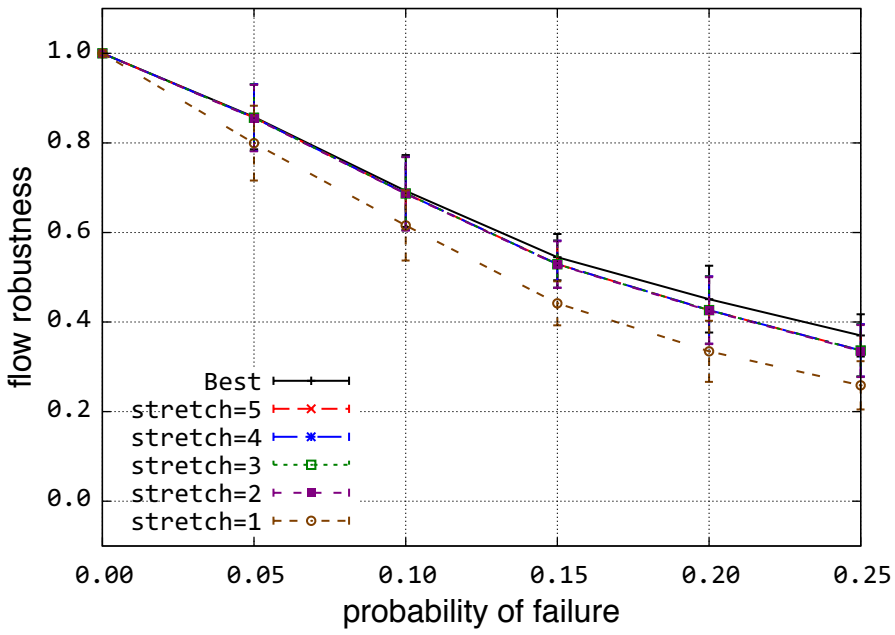

Figure E.132: Flow robustness vs. node & link failure probability for various stretch limits on the Tiscali logical topology

E.2.9 Verio logical

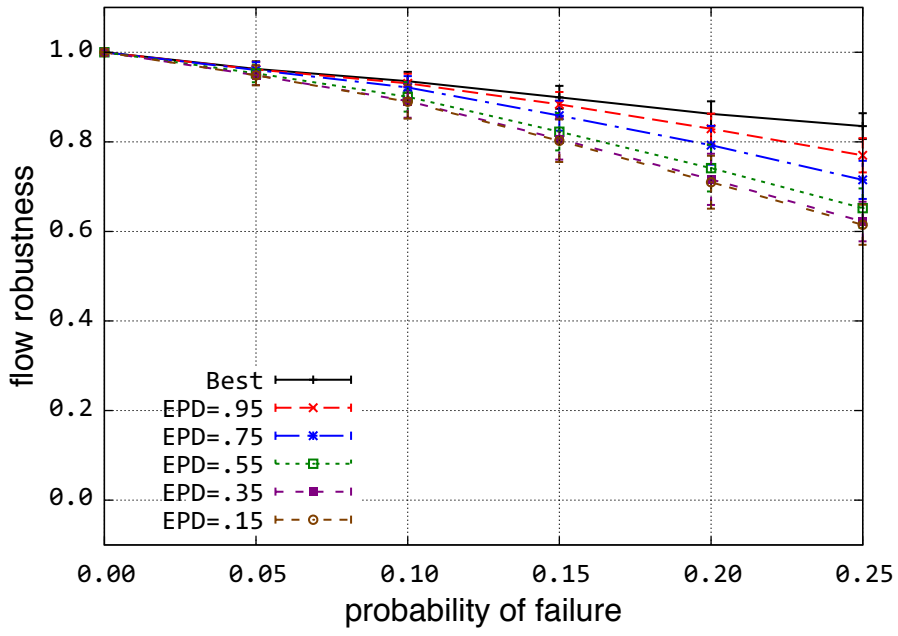

Figure E.133: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Verio logical topology

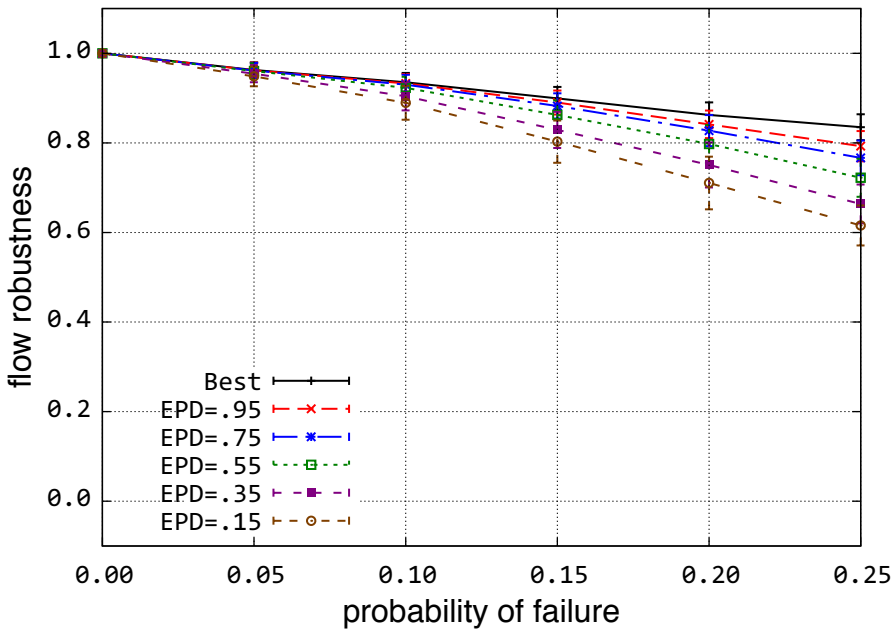

Figure E.134: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Verio logical topology

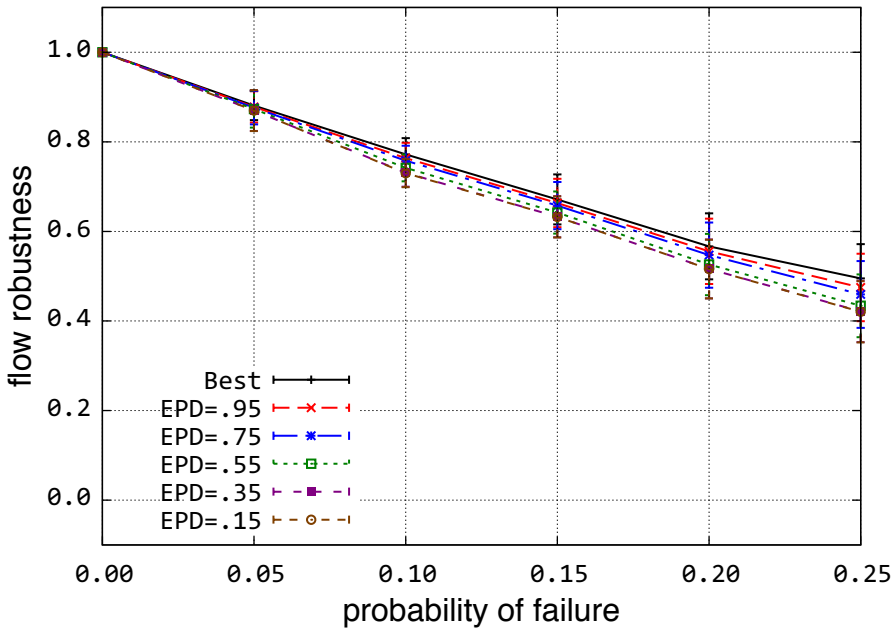

Figure E.135: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Verio logical topology

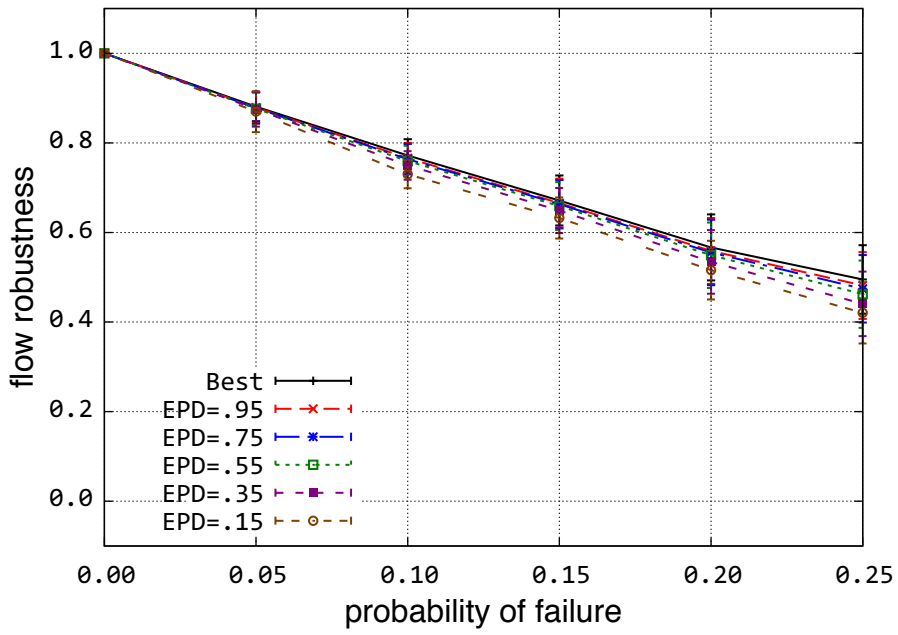

Figure E.136: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Verio logical topology

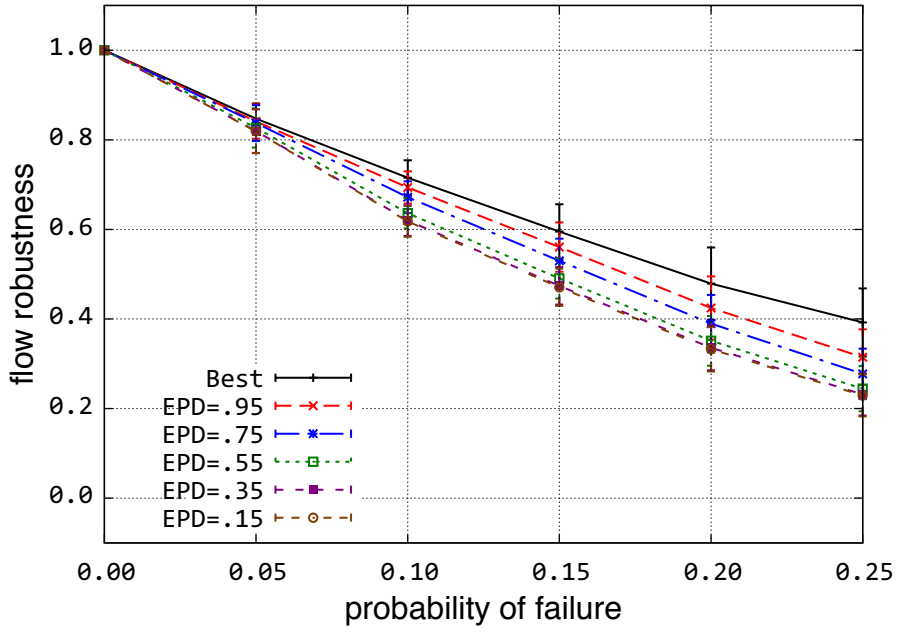

Figure E.137: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Verio logical topology

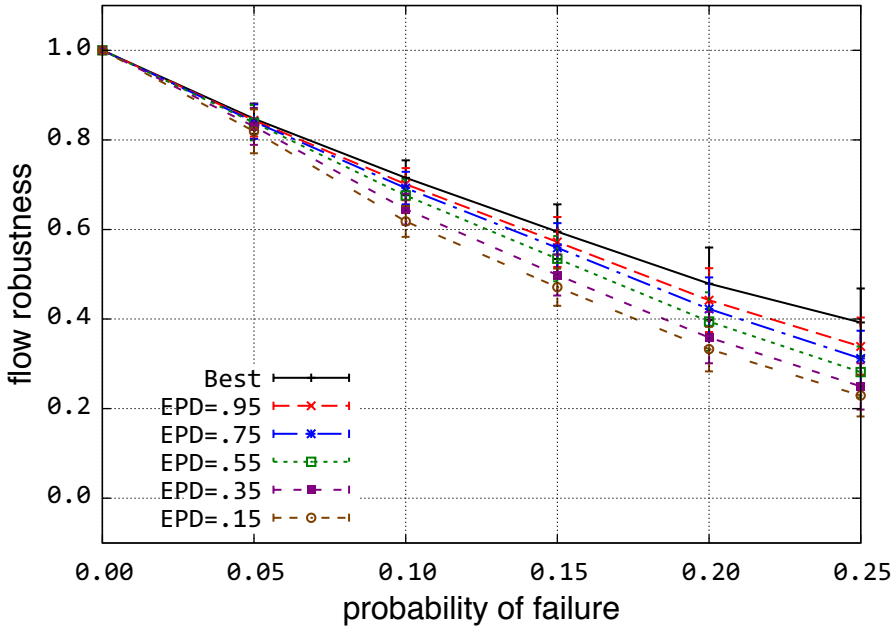

Figure E.138: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Verio logical topology

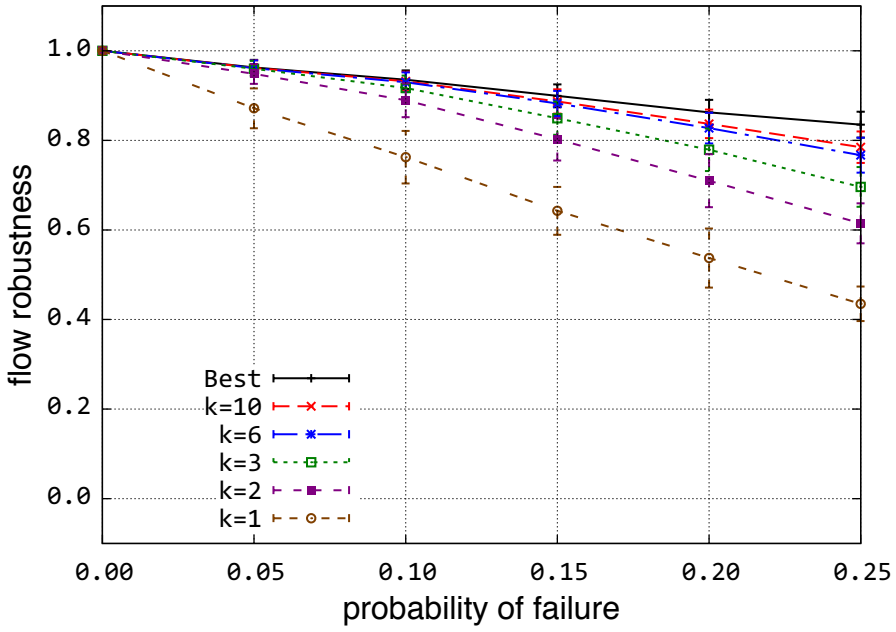

Figure E.139: Flow robustness vs. link failure probability for the Verio logical topology

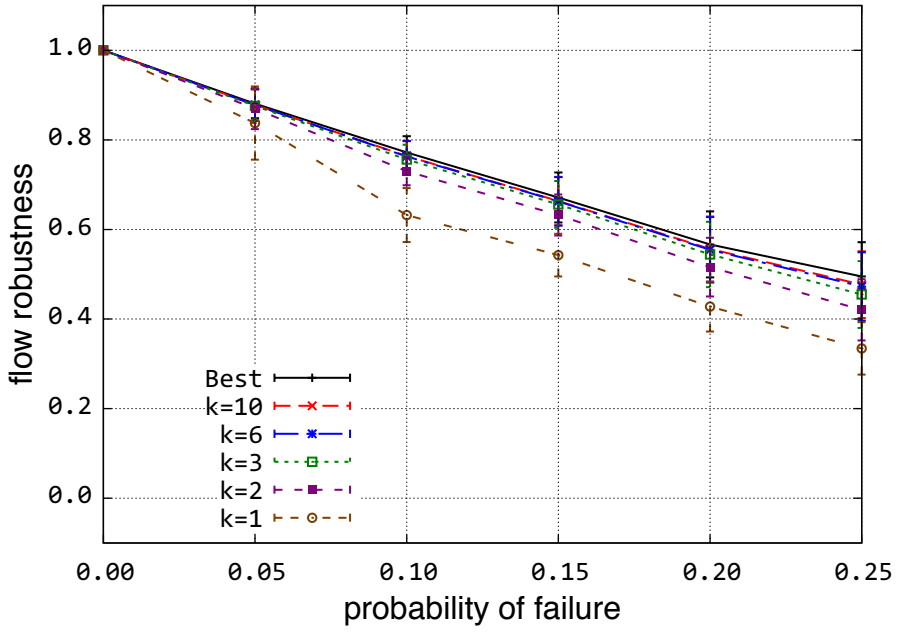

Figure E.140: Flow robustness vs. node failure probability for the Verio logical topology

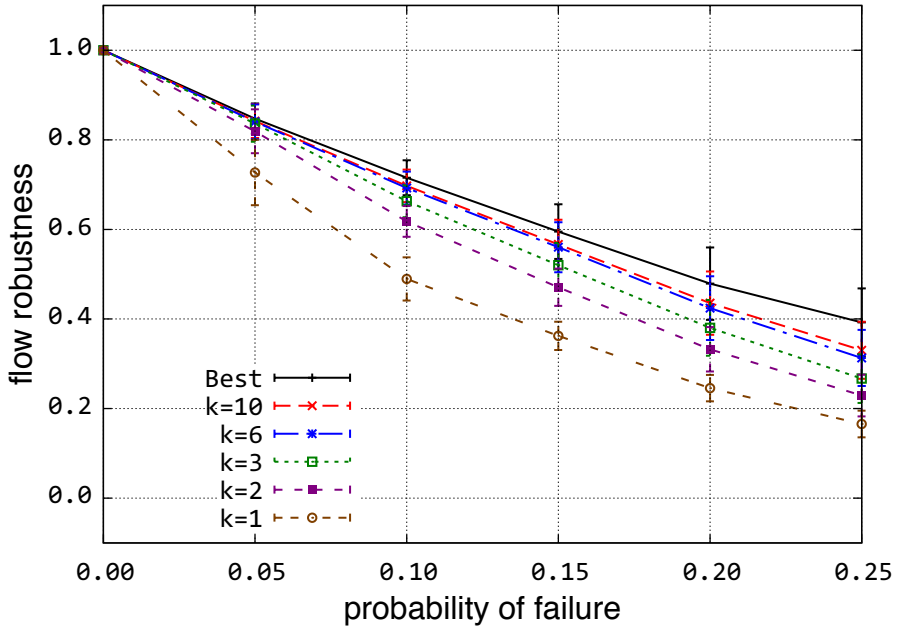

Figure E.141: Flow robustness vs. node & link failure probability for the Verio logical topology

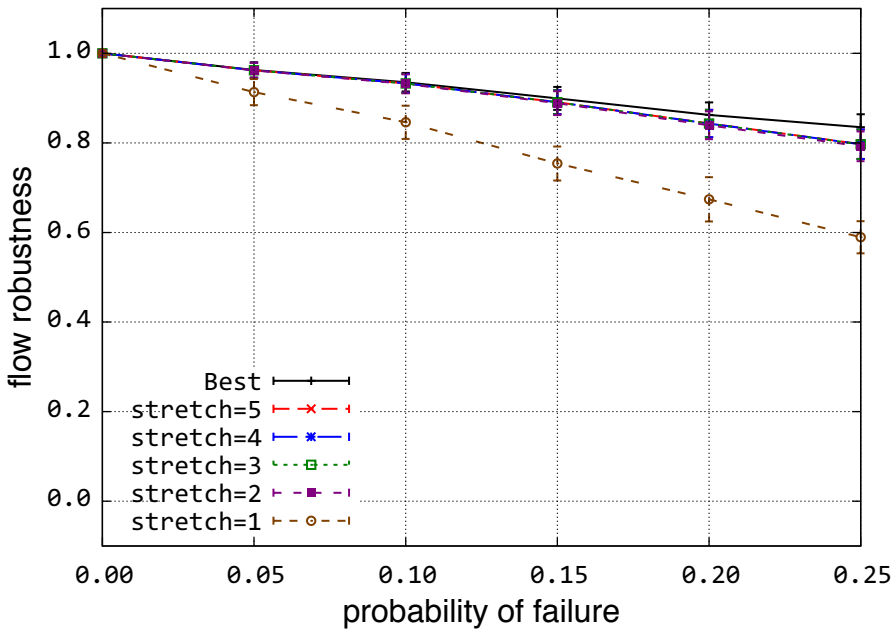

Figure E.142: Flow robustness vs. link failure probability for various stretch limits on the Verio logical topology

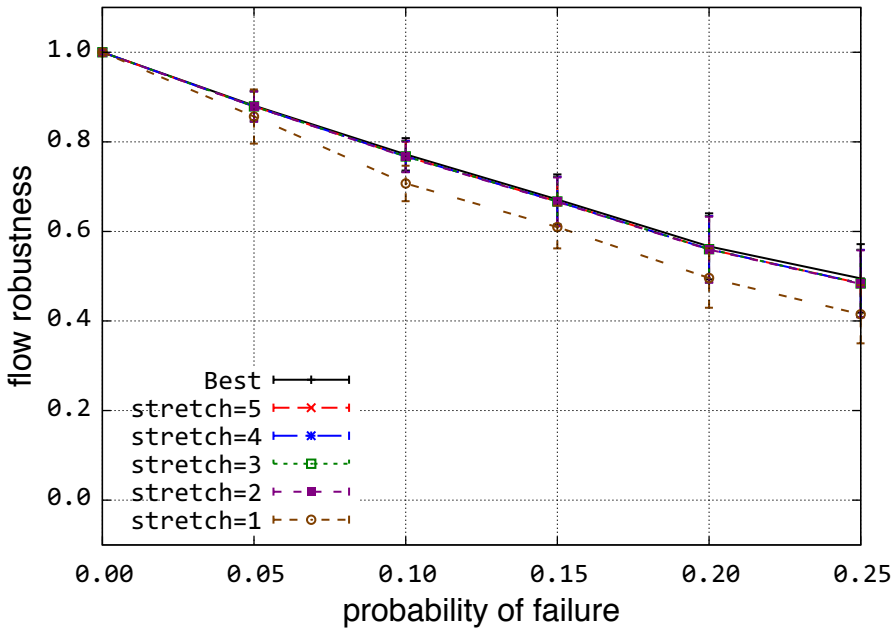

Figure E.143: Flow robustness vs. node failure probability for various stretch limits on the Verio logical topology

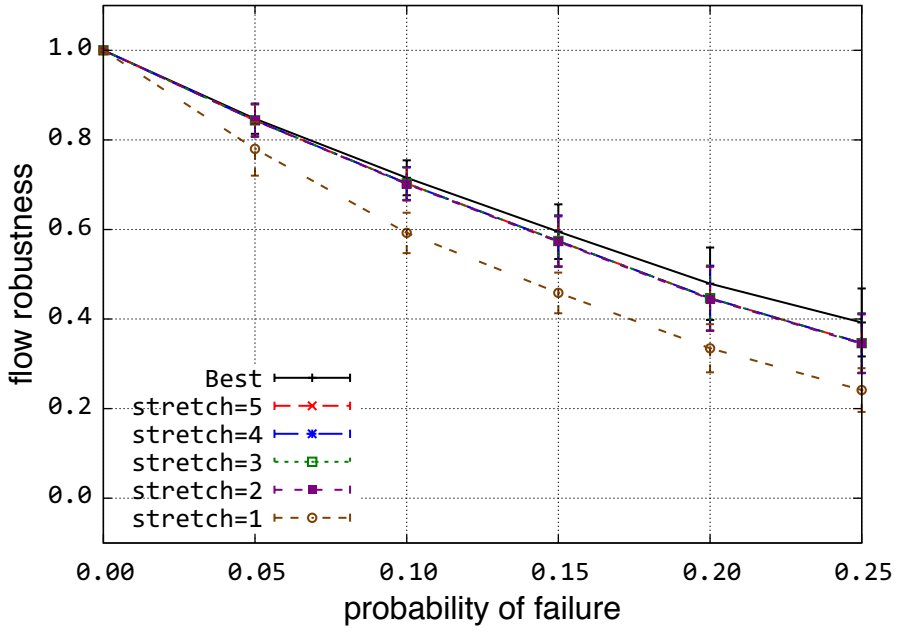

Figure E.144: Flow robustness vs. node & link failure probability for various stretch limits on the Verio logical topology

E.2.10 VSNL logical

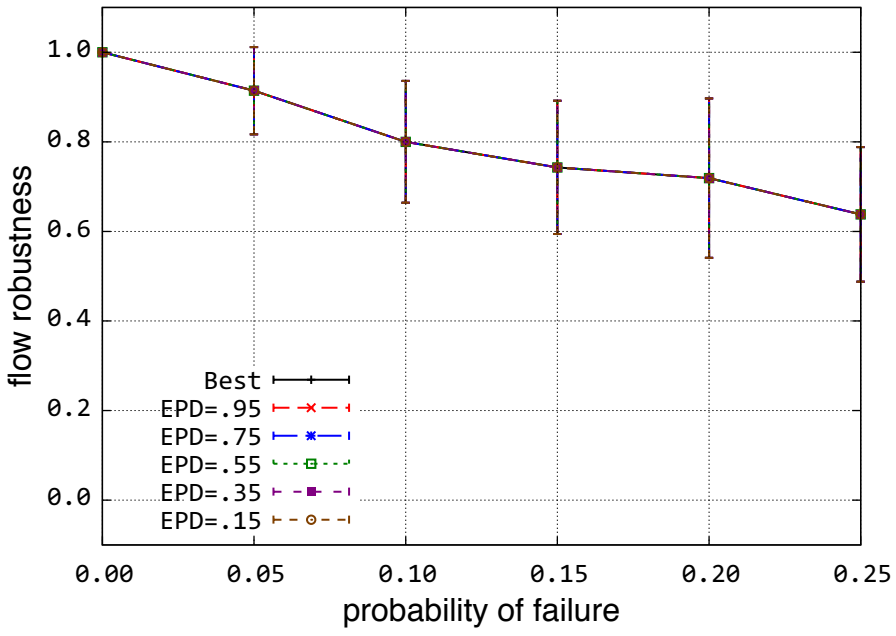

Figure E.145: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the VSNL logical topology

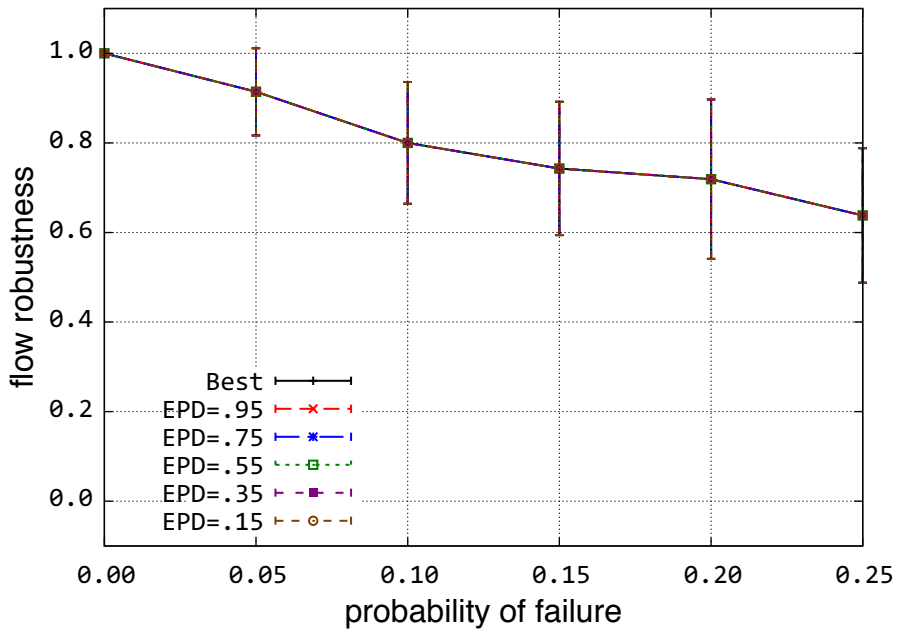

Figure E.146: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the VSNL logical topology

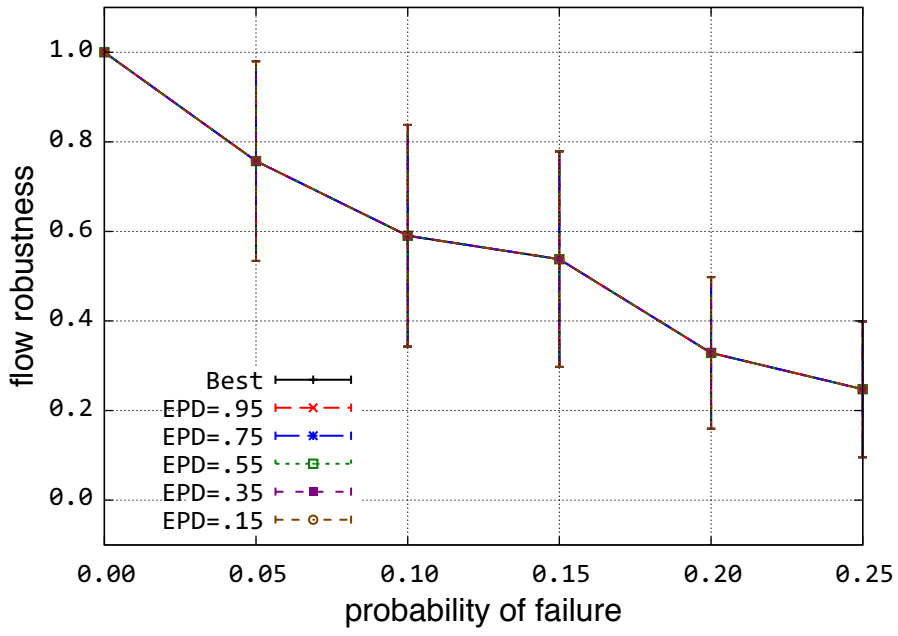

Figure E.147: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the VSNL logical topology

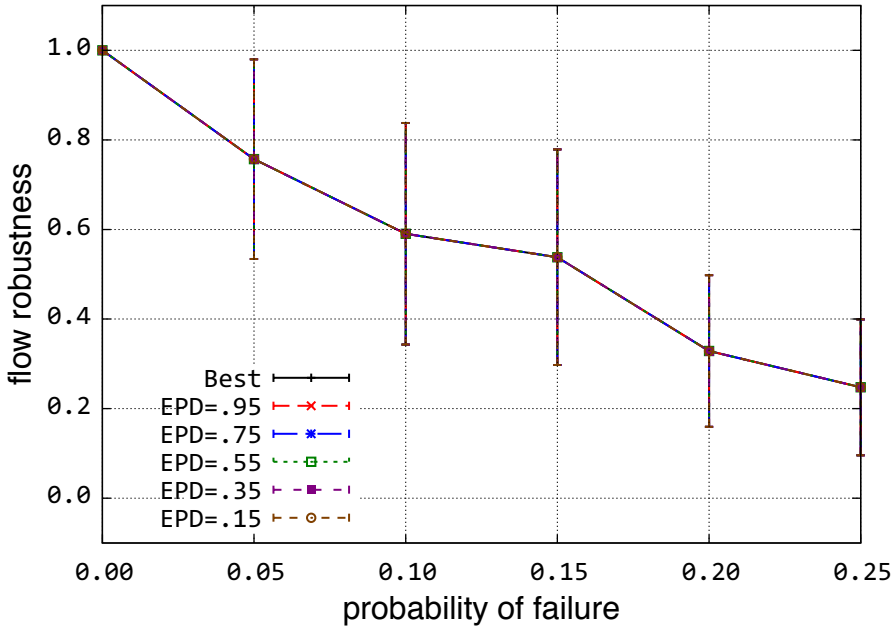

Figure E.148: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the VSNL logical topology

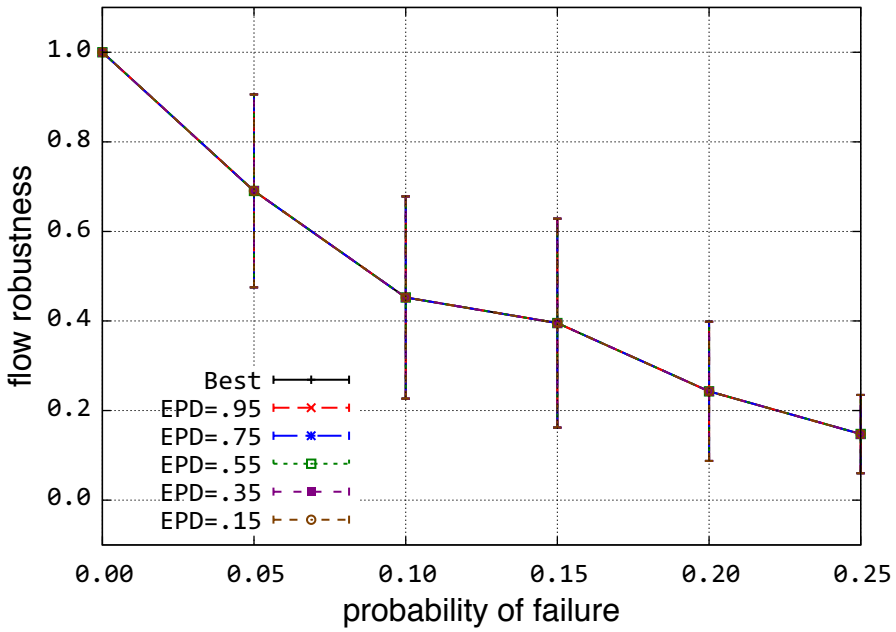

Figure E.149: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the VSNL logical topology

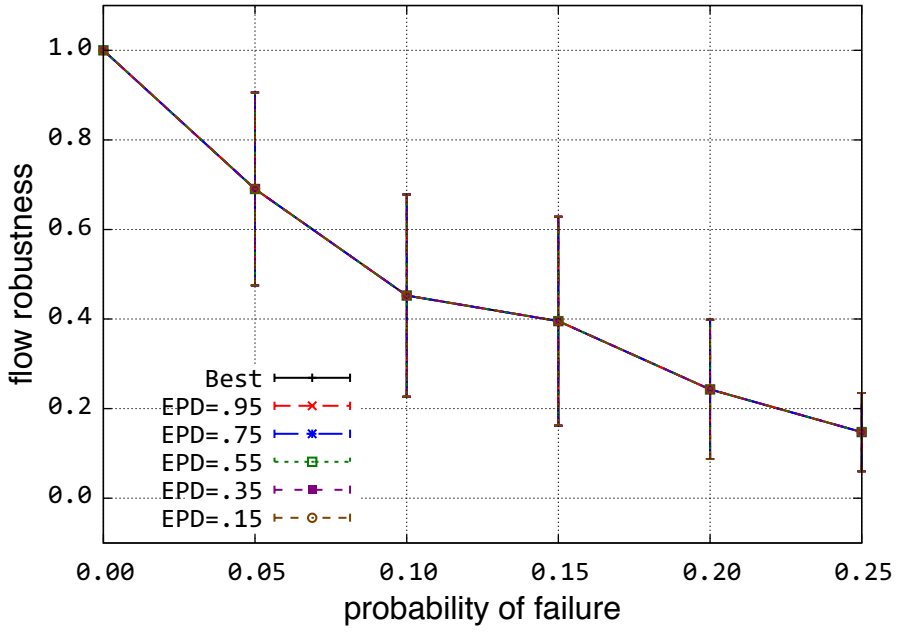

Figure E.150: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the VSNL logical topology

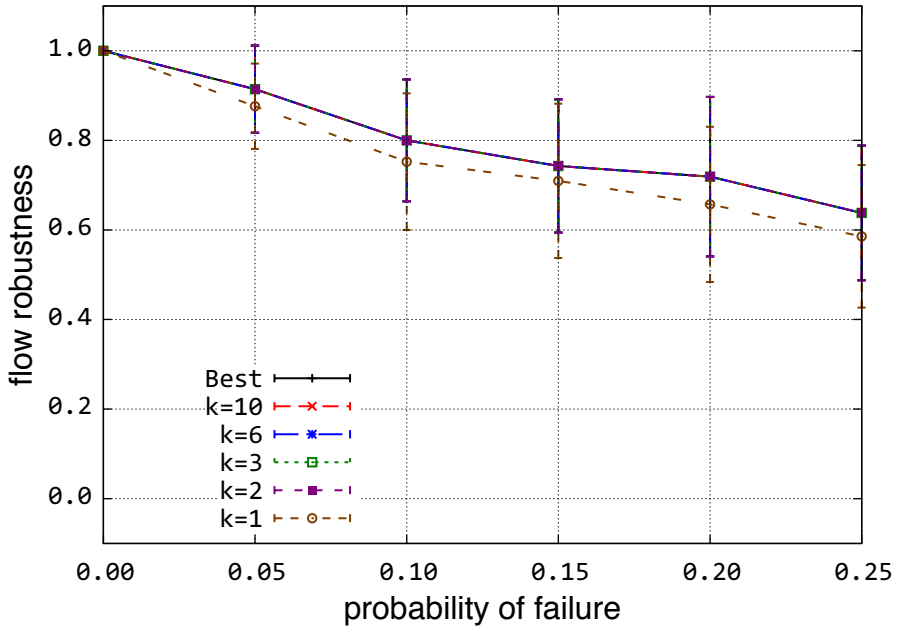

Figure E.151: Flow robustness vs. link failure probability for the VSNL logical topology

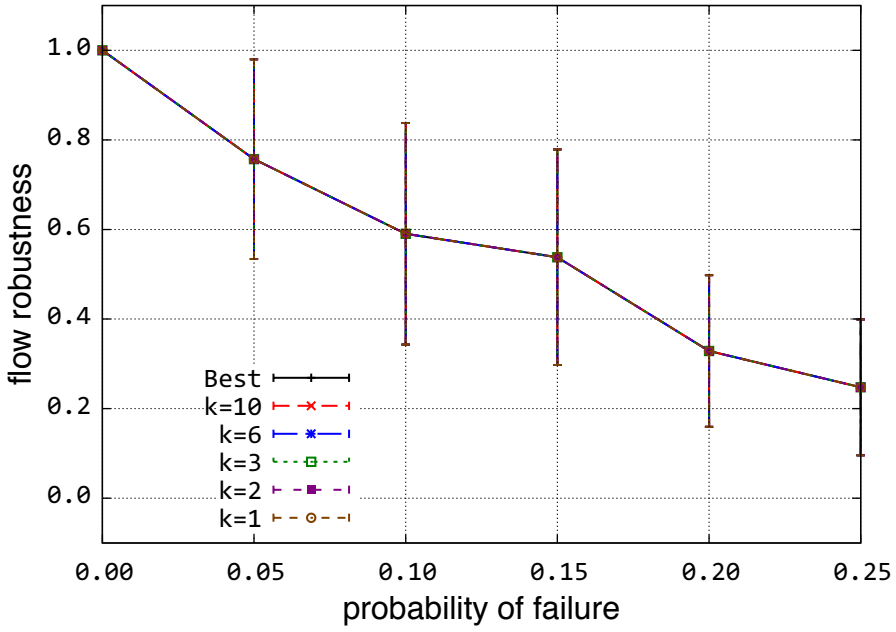

Figure E.152: Flow robustness vs. node failure probability for the VSNL logical topology

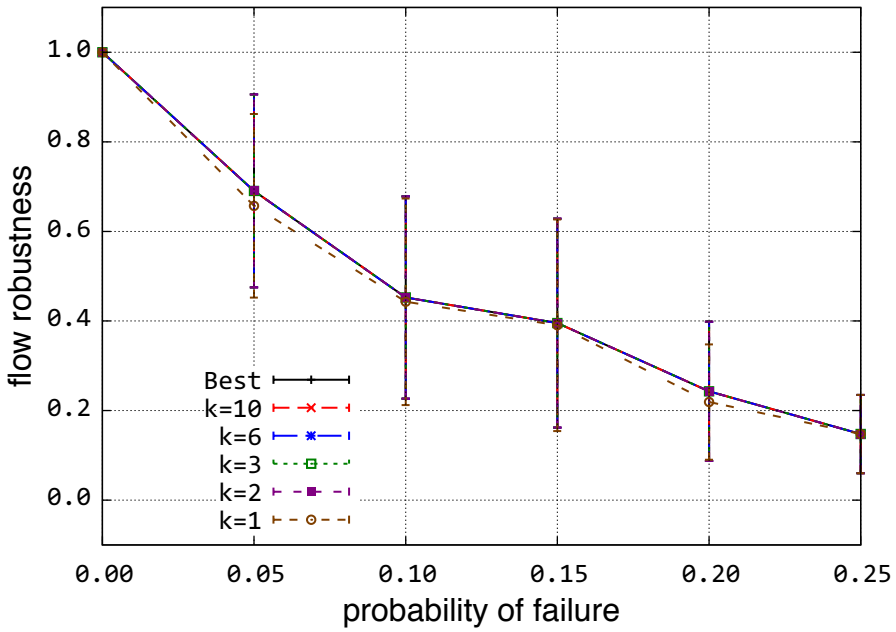

Figure E.153: Flow robustness vs. node & link failure probability for the VSNL logical topology

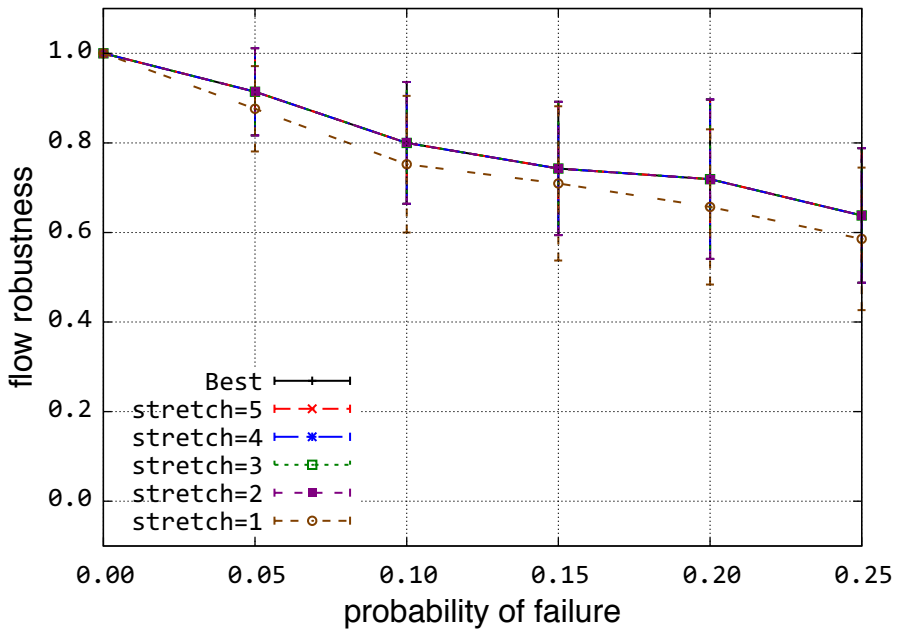

Figure E.154: Flow robustness vs. link failure probability for various stretch limits on the VSNL logical topology

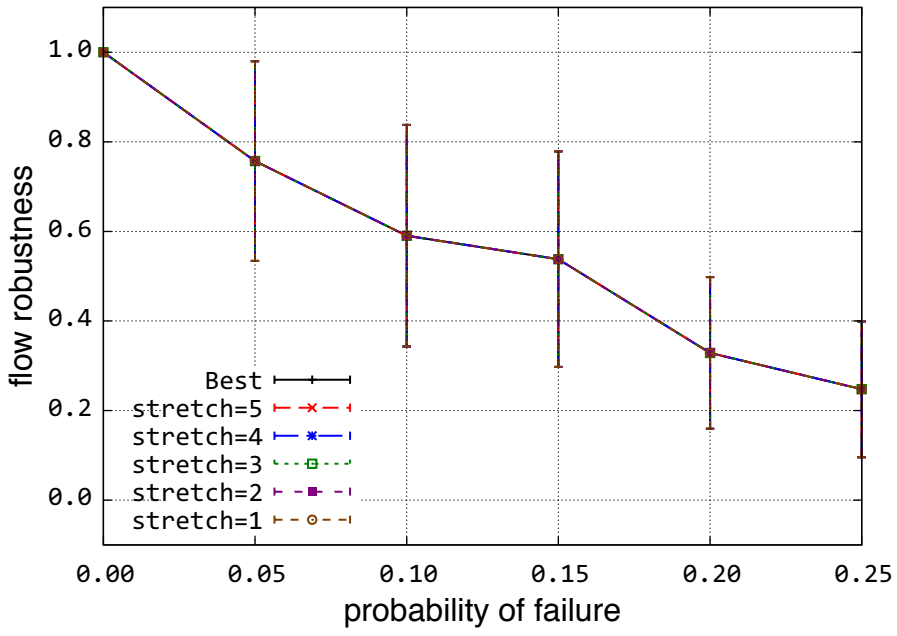

Figure E.155: Flow robustness vs. node failure probability for various stretch limits on the VSNL logical topology

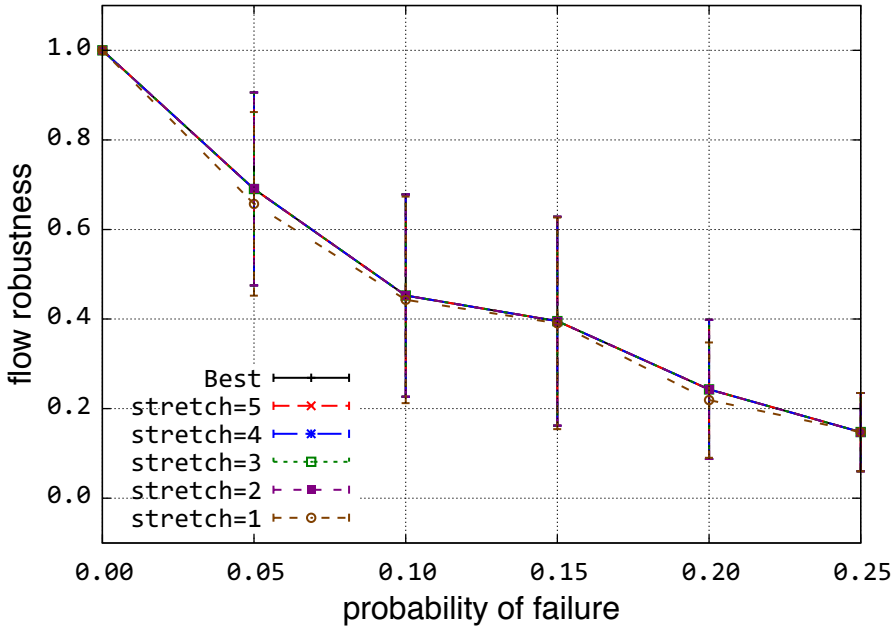

Figure E.156: Flow robustness vs. node & link failure probability for various stretch limits on the VSNL logical topology

E.3 Synthetic Topologies

E.3.1 Full-mesh

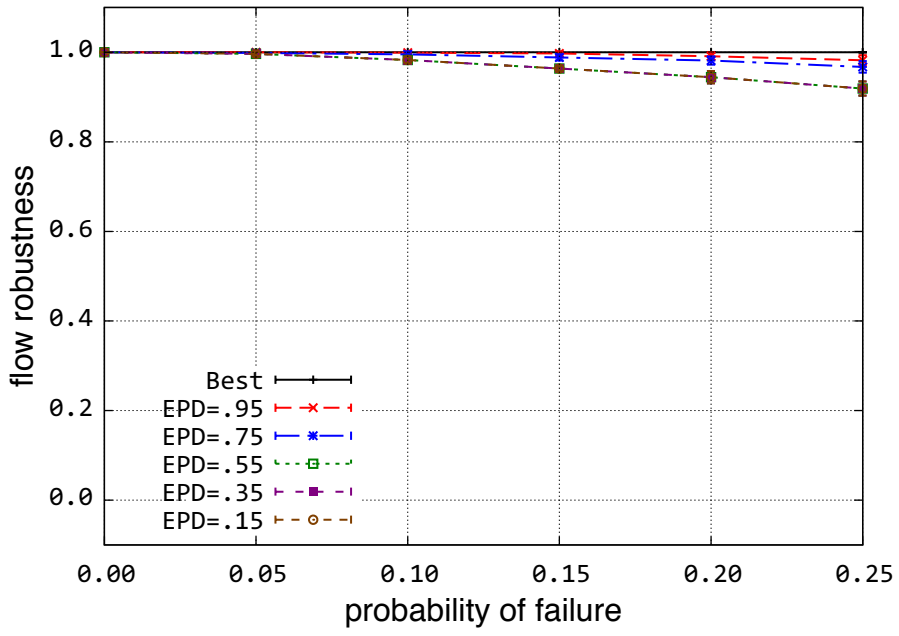

Figure E.157: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Full-mesh topology

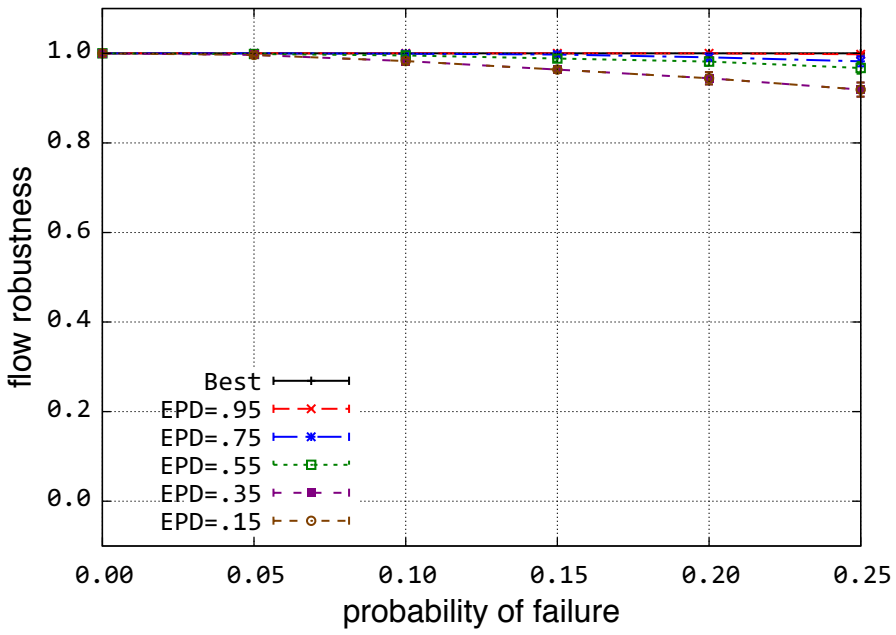

Figure E.158: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Full-mesh topology

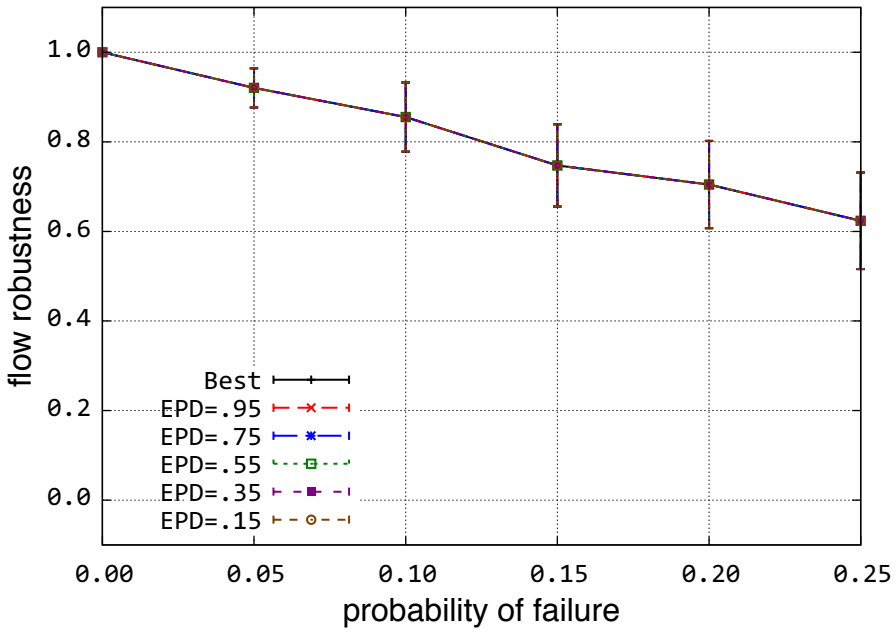

Figure E.159: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Full-mesh topology

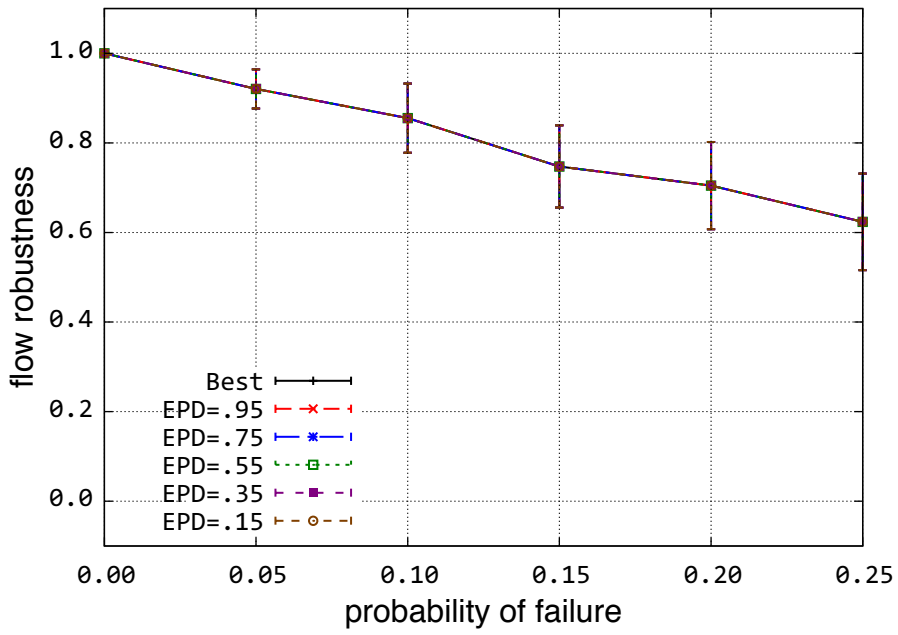

Figure E.160: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Full-mesh topology

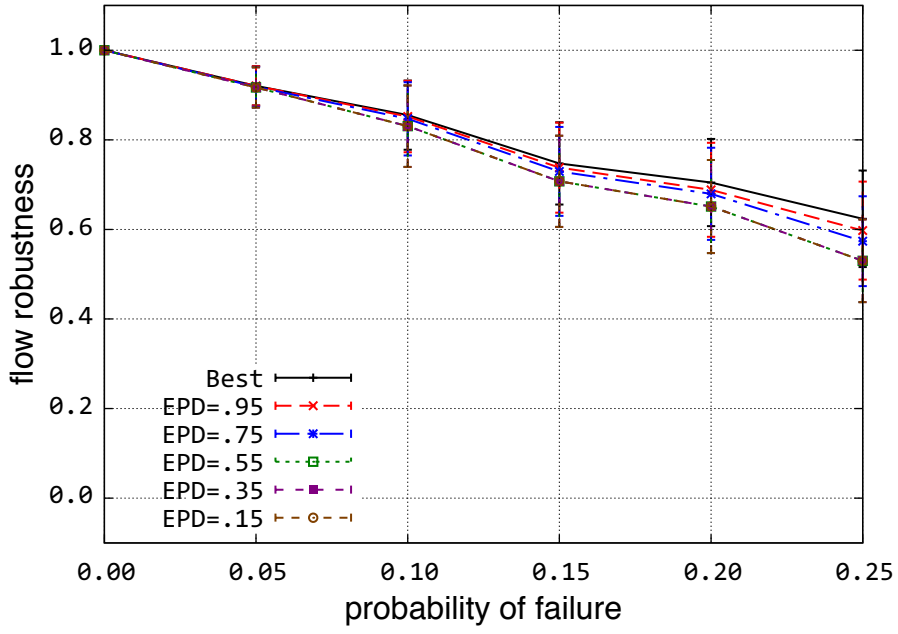

Figure E.161: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Full-mesh topology

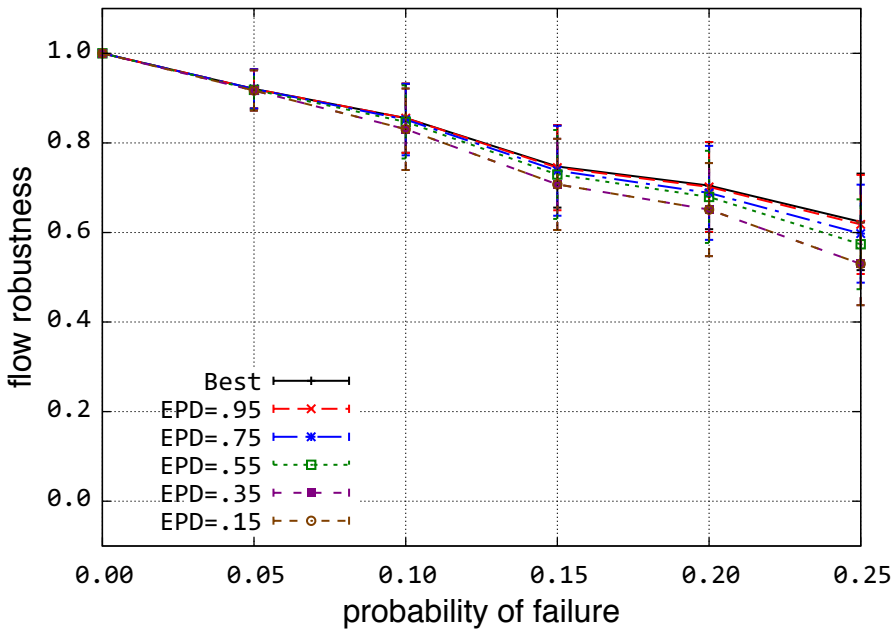

Figure E.162: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Full-mesh topology

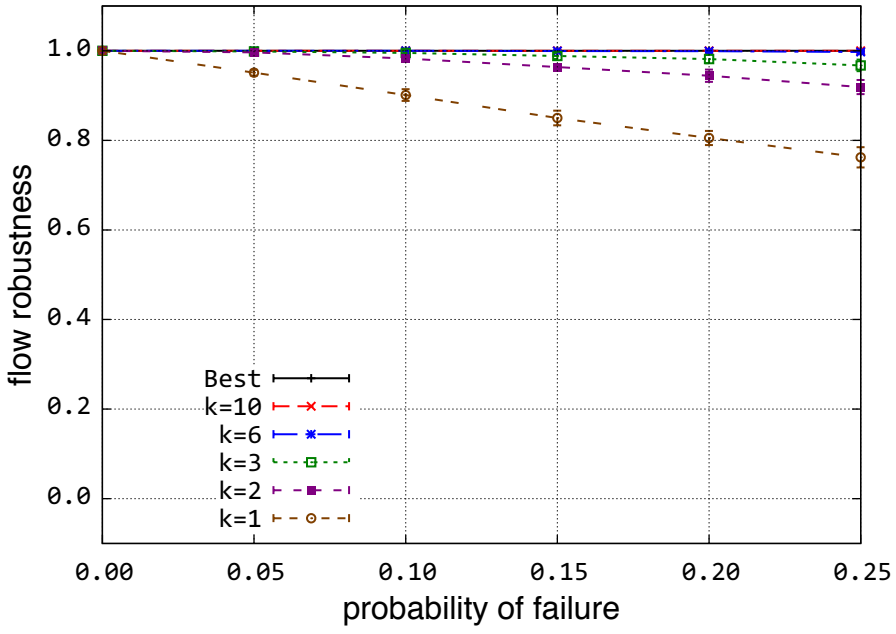

Figure E.163: Flow robustness vs. link failure probability for the Full-mesh topology

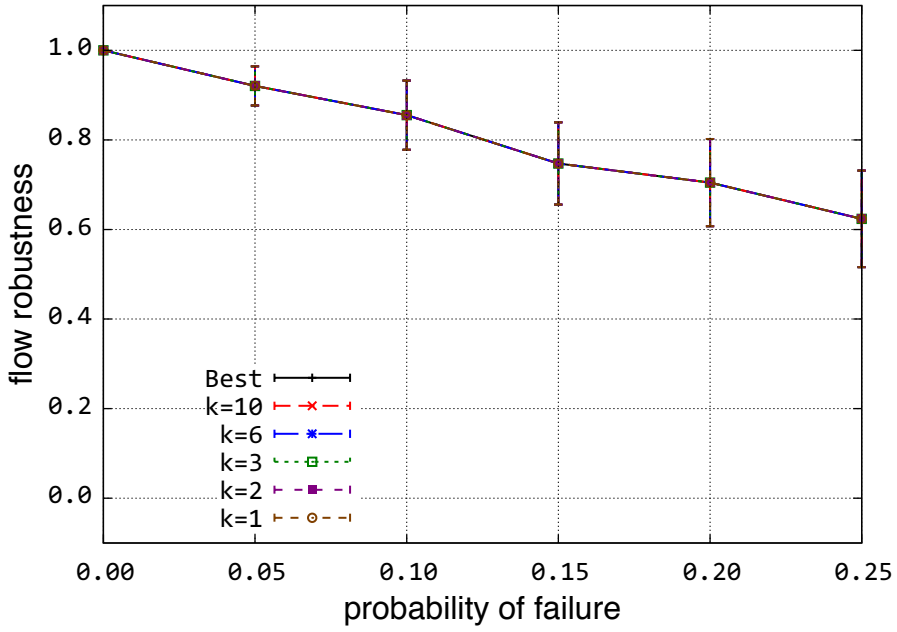

Figure E.164: Flow robustness vs. node failure probability for the Full-mesh topology

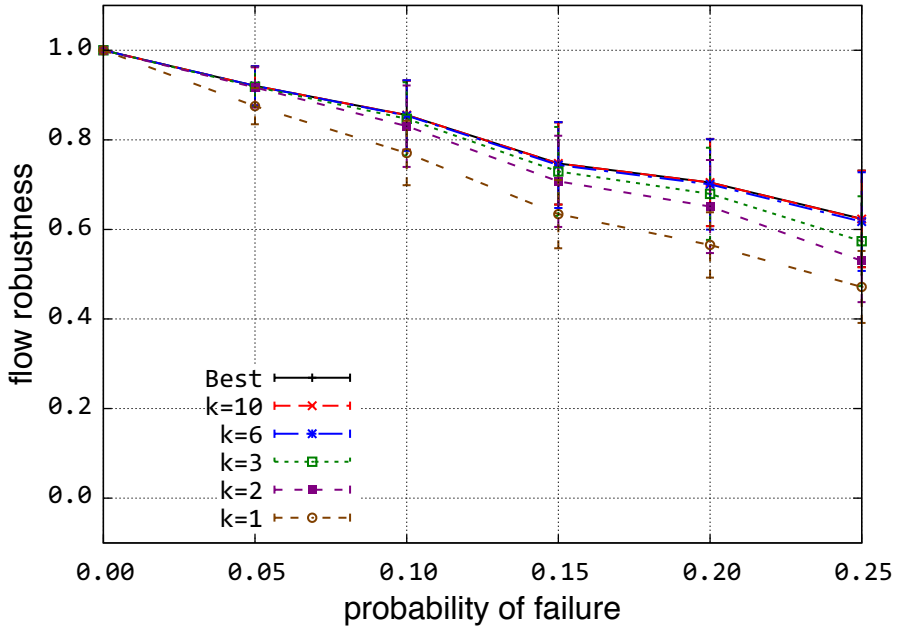

Figure E.165: Flow robustness vs. node & link failure probability for the Full-mesh topology

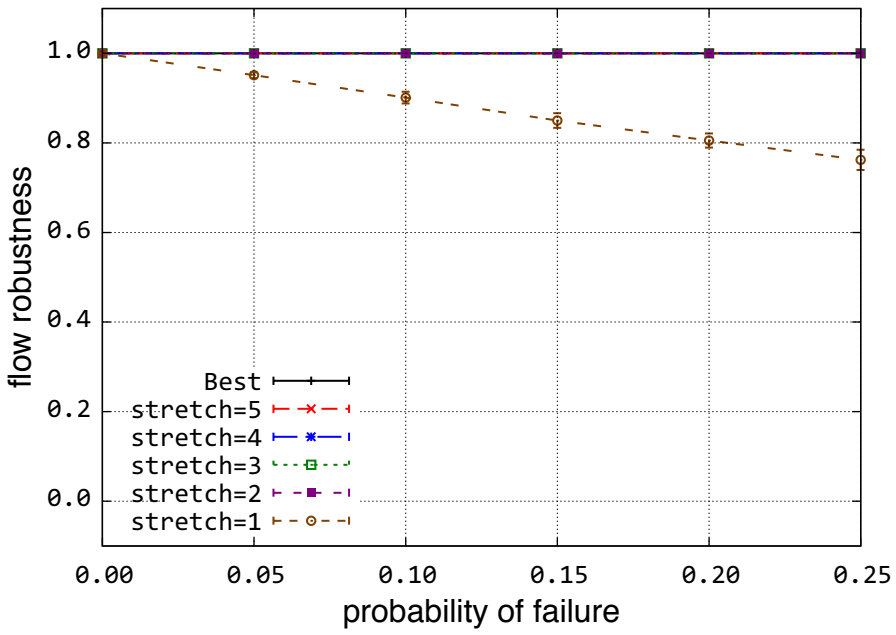

Figure E.166: Flow robustness vs. link failure probability for various stretch limits on the Full-mesh topology

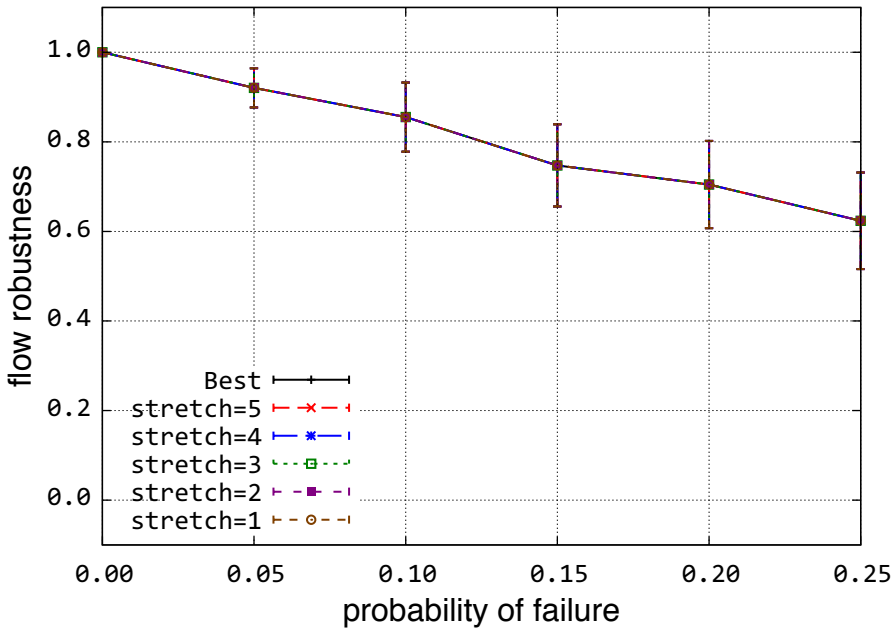

Figure E.167: Flow robustness vs. node failure probability for various stretch limits on the Full-mesh topology

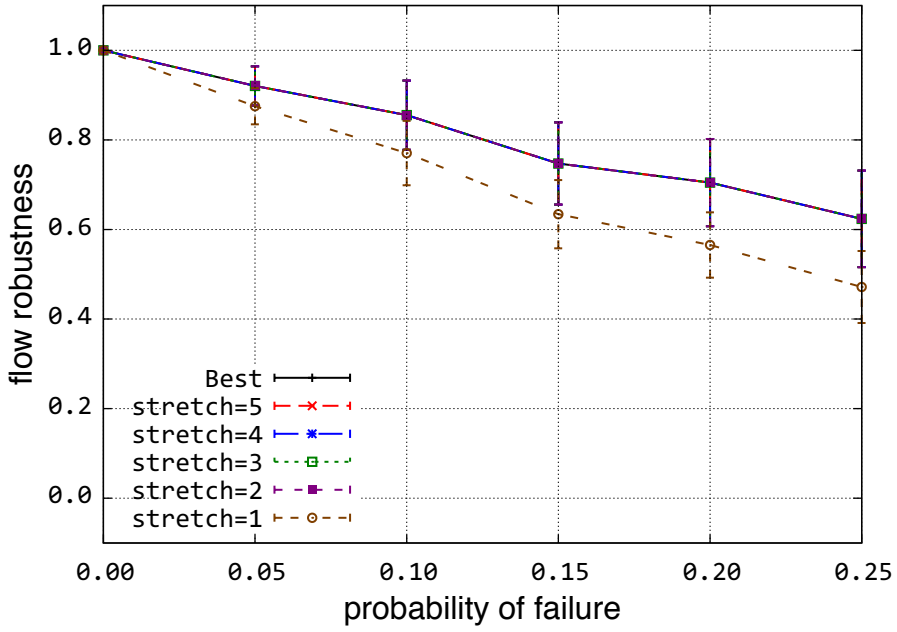

Figure E.168: Flow robustness vs. node & link failure probability for various stretch limits on the Full-mesh topology

E.3.2 Manhattan grid

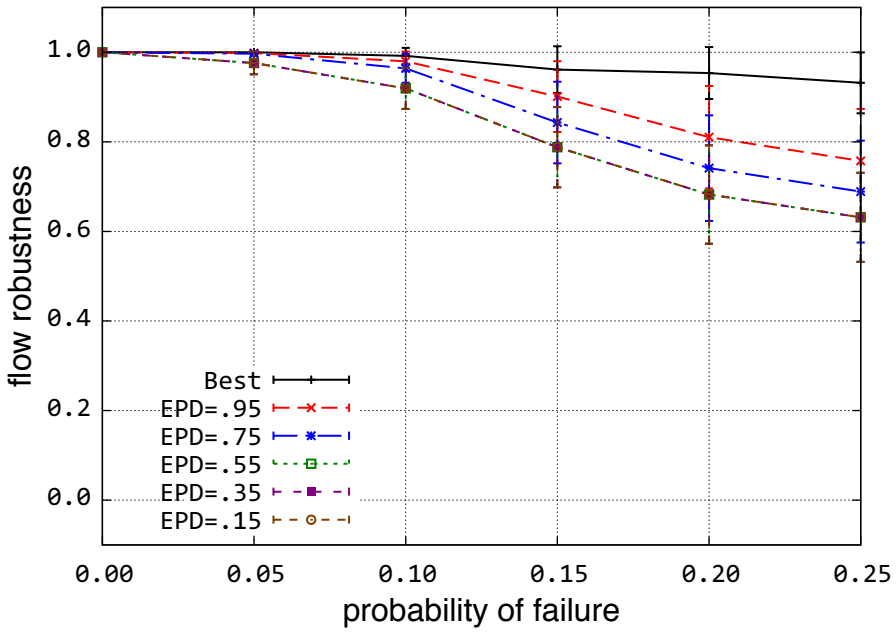

Figure E.169: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Manhattan grid topology

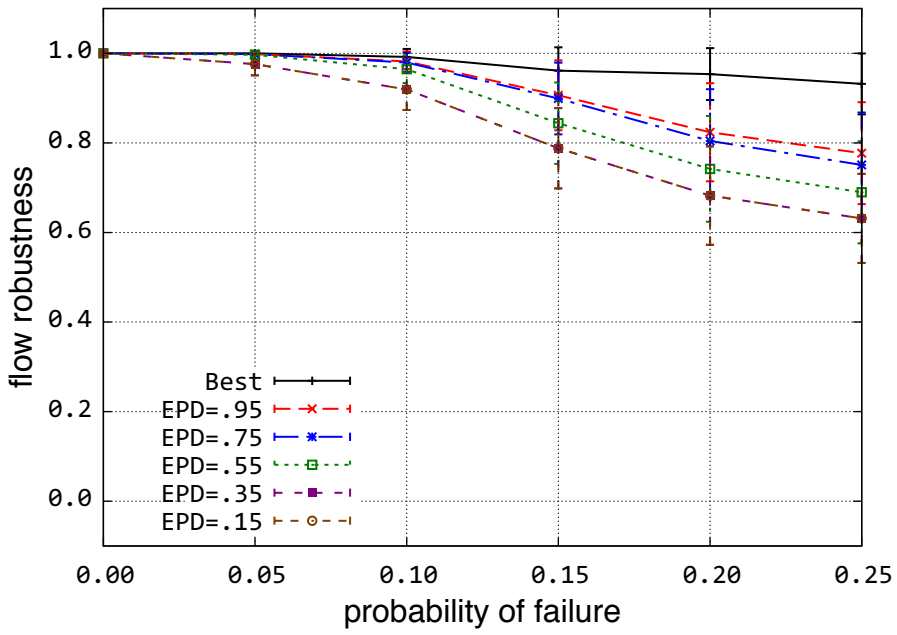

Figure E.170: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Manhattan grid topology

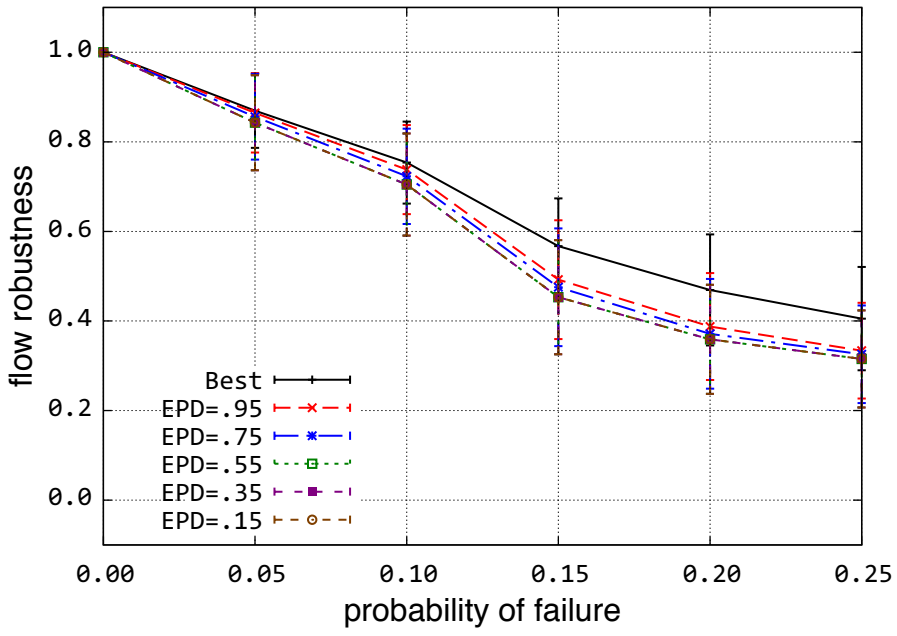

Figure E.171: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Manhattan grid topology

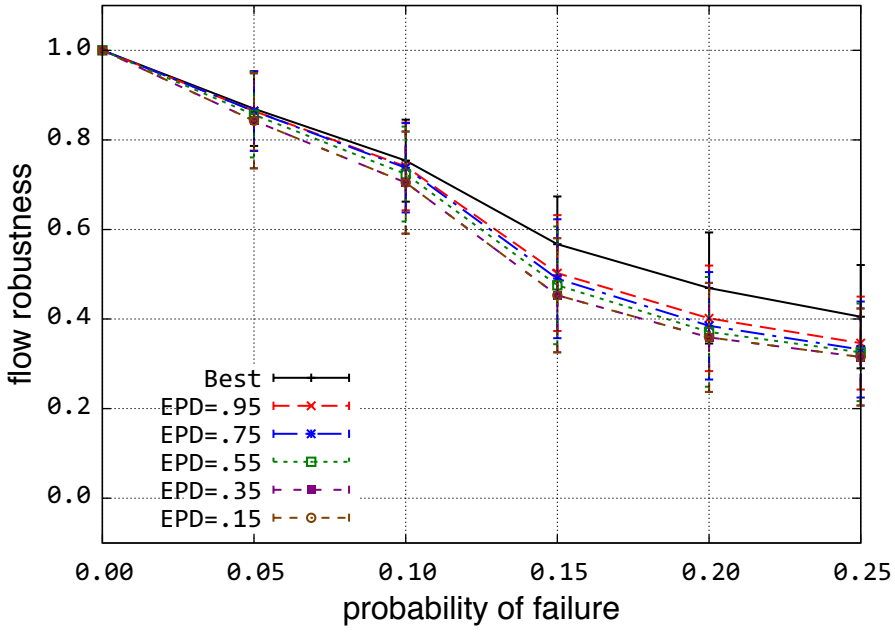

Figure E.172: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Manhattan grid topology

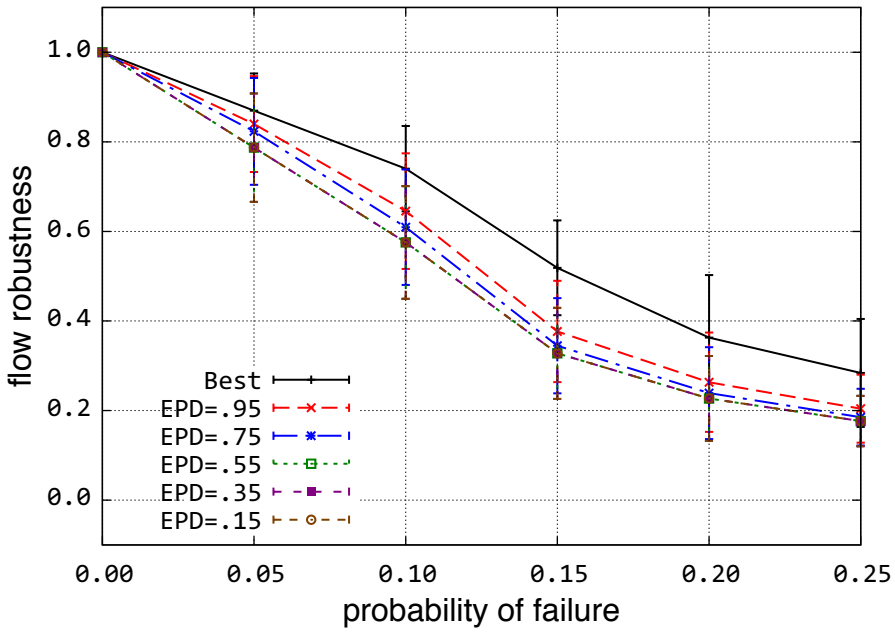

Figure E.173: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Manhattan grid topology

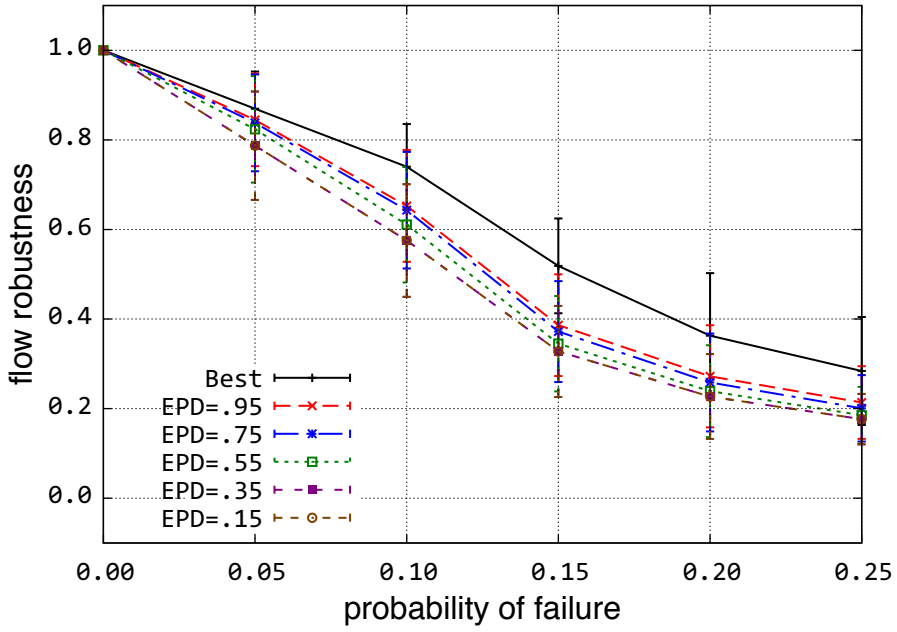

Figure E.174: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Manhattan grid topology

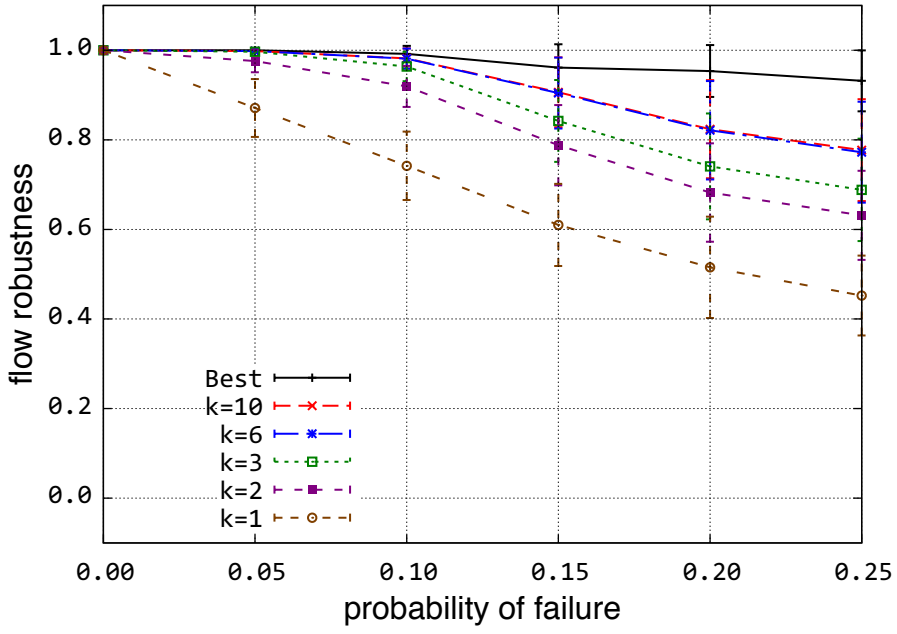

Figure E.175: Flow robustness vs. link failure probability for the Manhattan grid topology

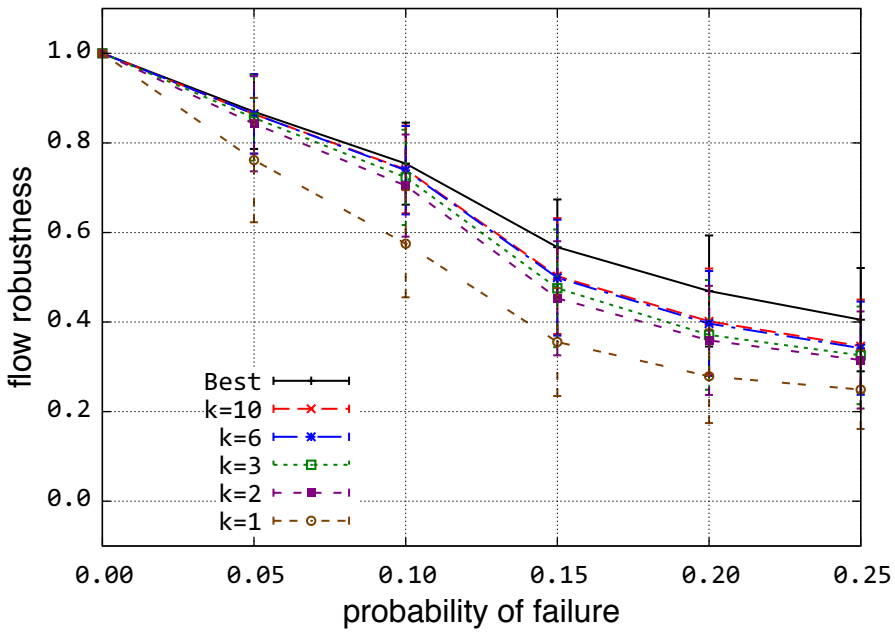

Figure E.176: Flow robustness vs. node failure probability for the Manhattan grid topology

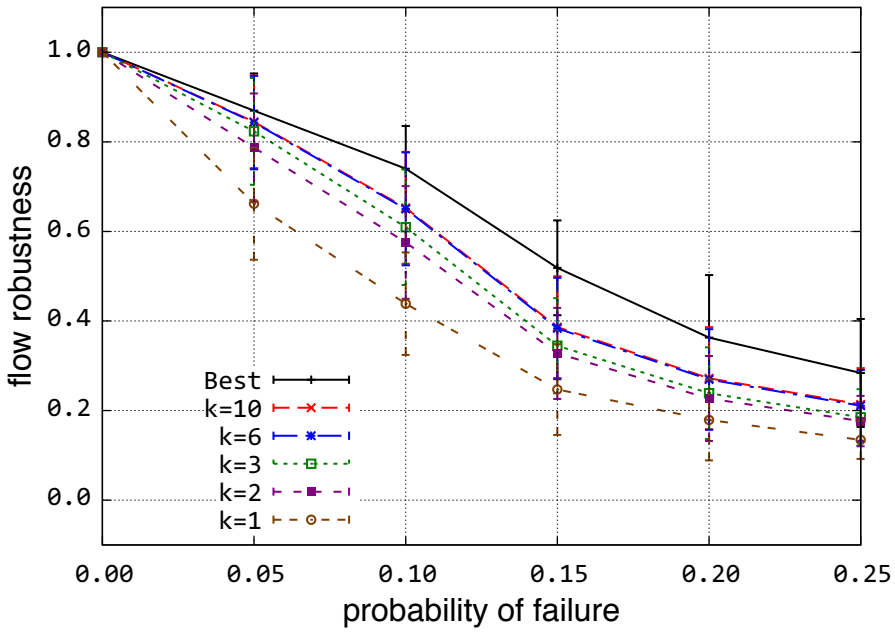

Figure E.177: Flow robustness vs. node & link failure probability for the Manhattan grid topology

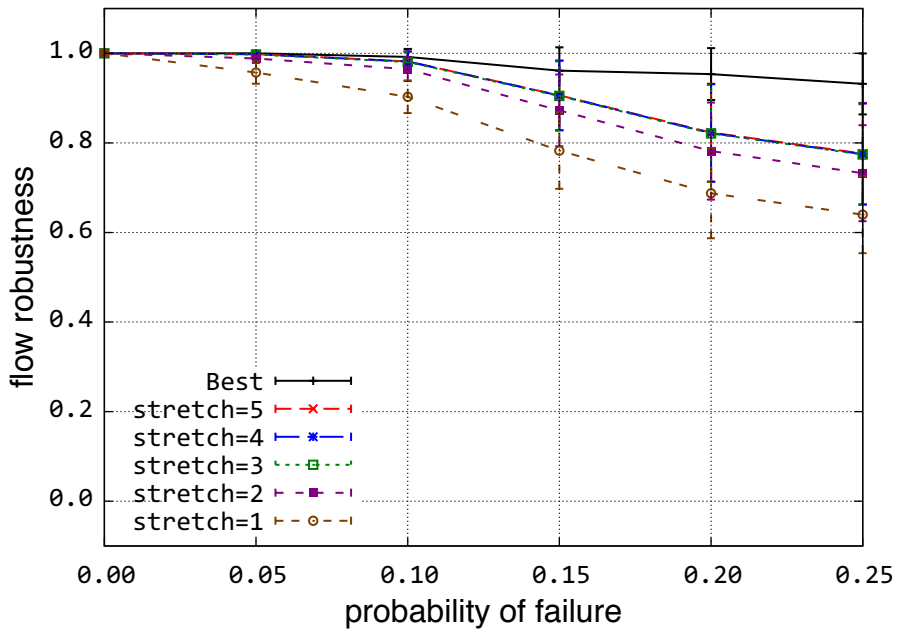

Figure E.178: Flow robustness vs. link failure probability for various stretch limits on the Manhattan grid topology

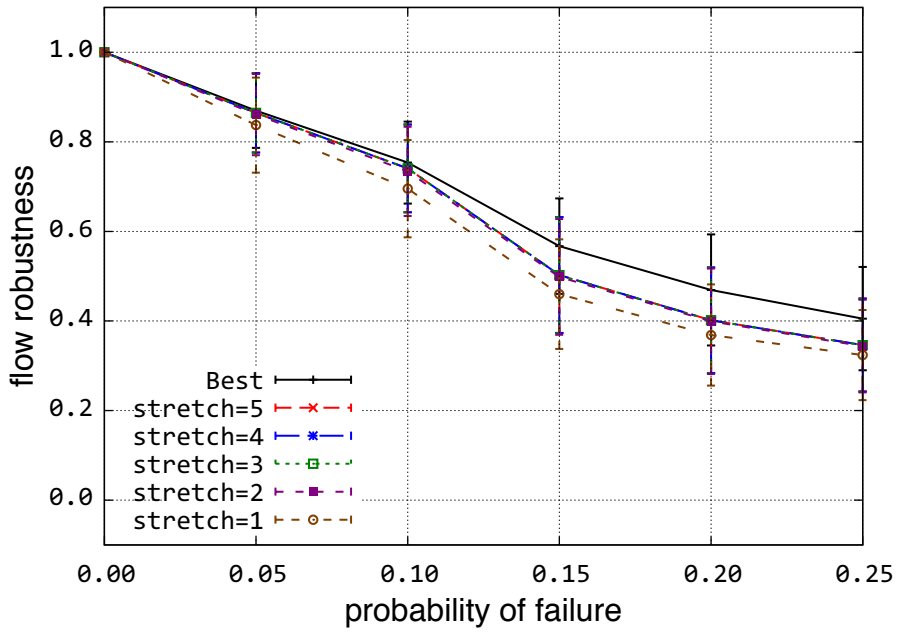

Figure E.179: Flow robustness vs. node failure probability for various stretch limits on the Manhattan grid topology

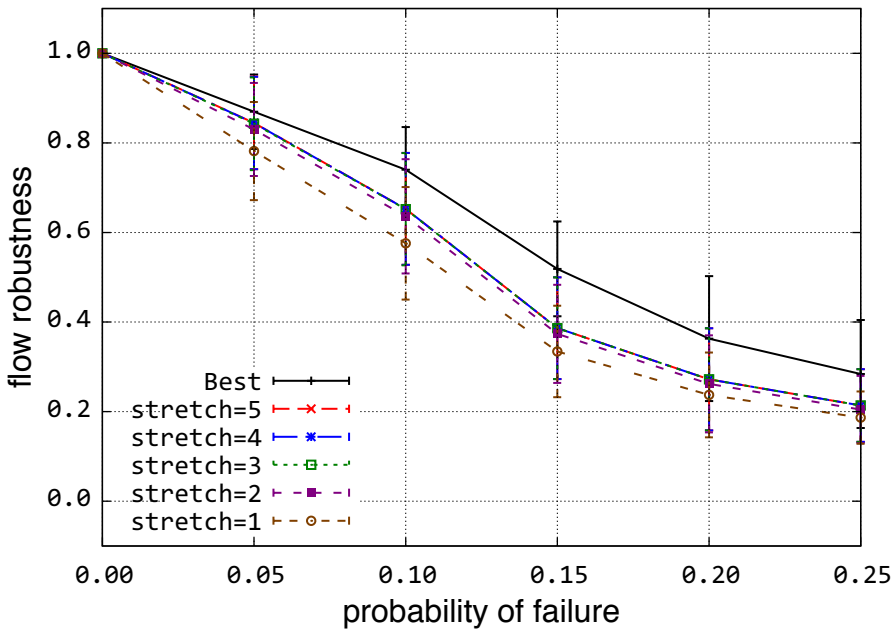

Figure E.180: Flow robustness vs. node & link failure probability for various stretch limits on the Manhattan grid topology

E.3.3 Ring

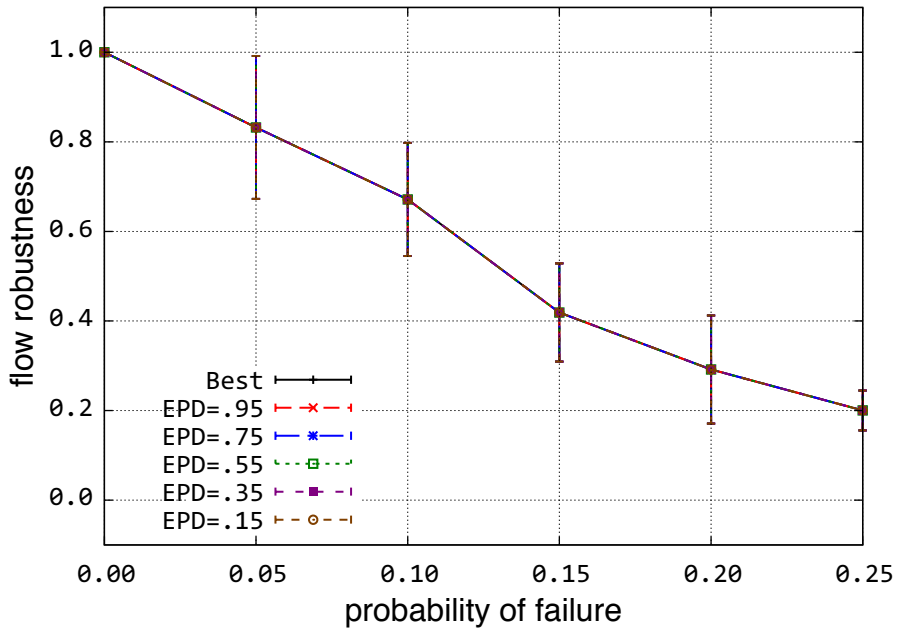

Figure E.181: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Ring topology

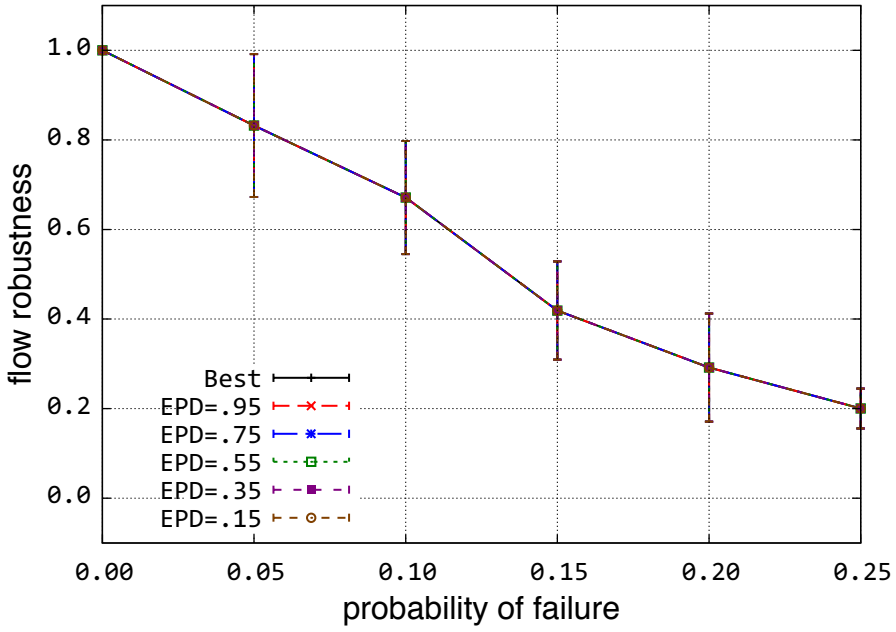

Figure E.182: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Ring topology

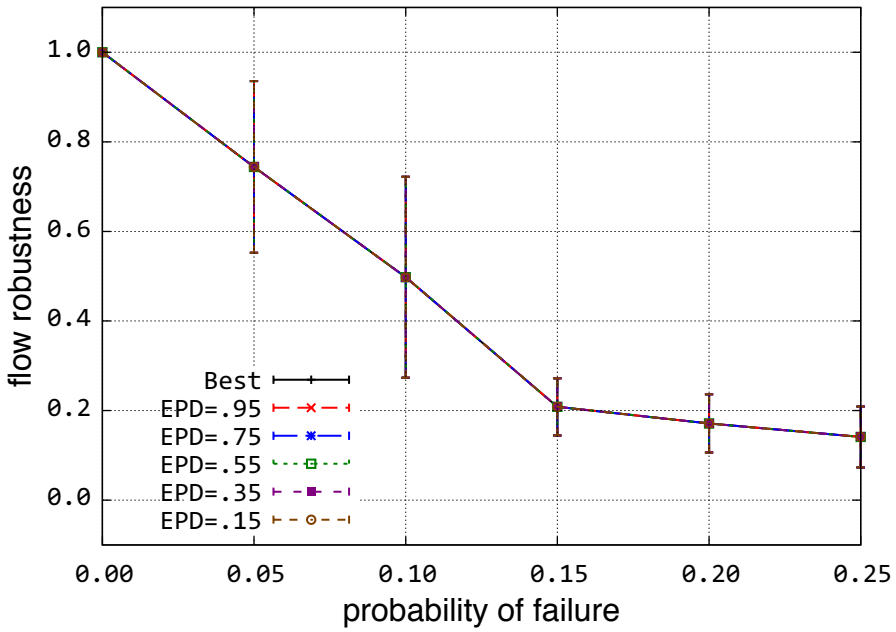

Figure E.183: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Ring topology

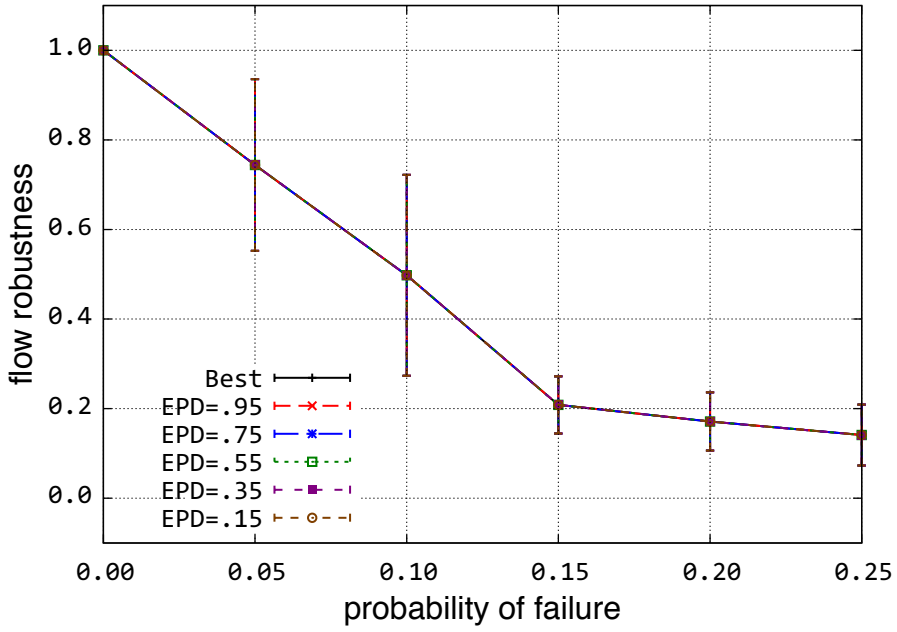

Figure E.184: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Ring topology

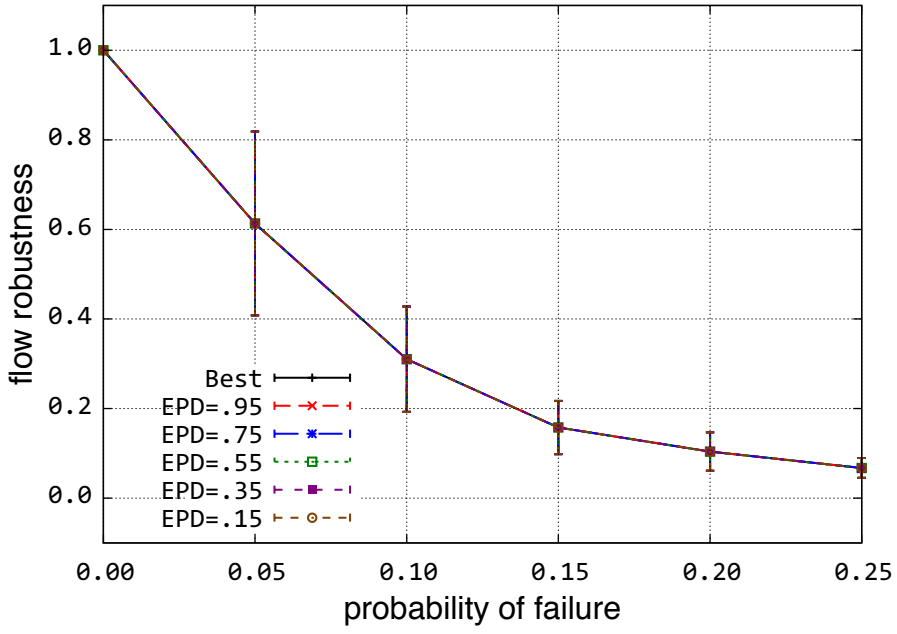

Figure E.185: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Ring topology

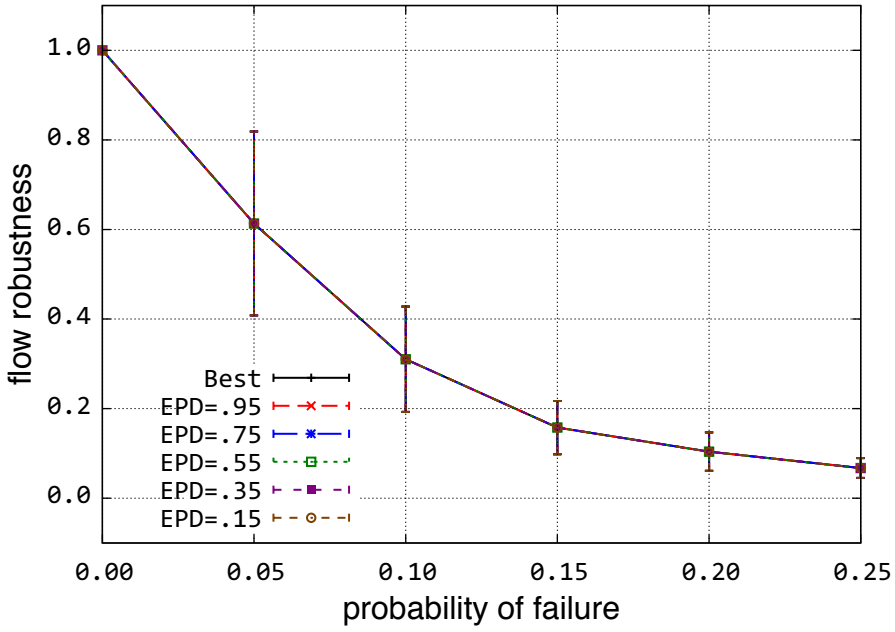

Figure E.186: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Ring topology

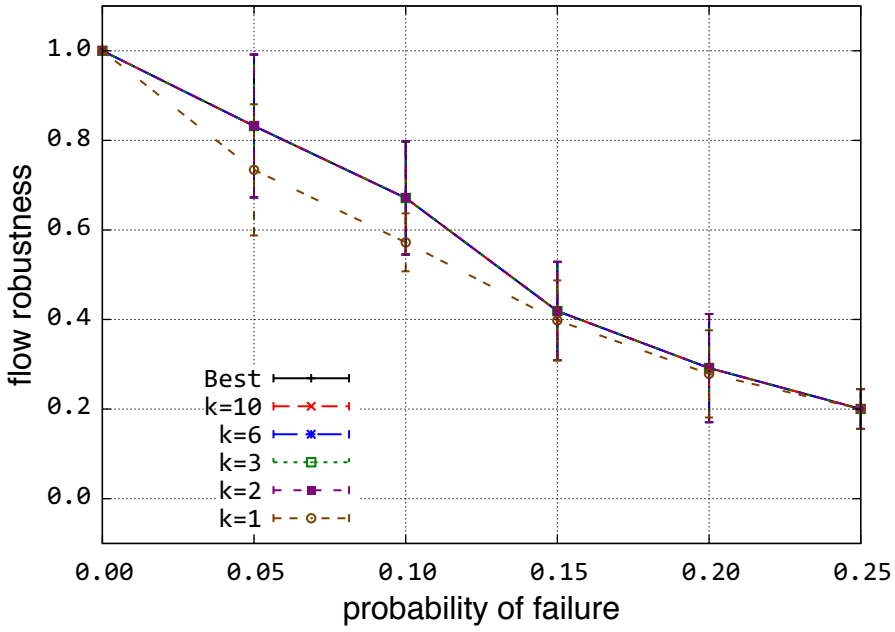

Figure E.187: Flow robustness vs. link failure probability for the Ring topology

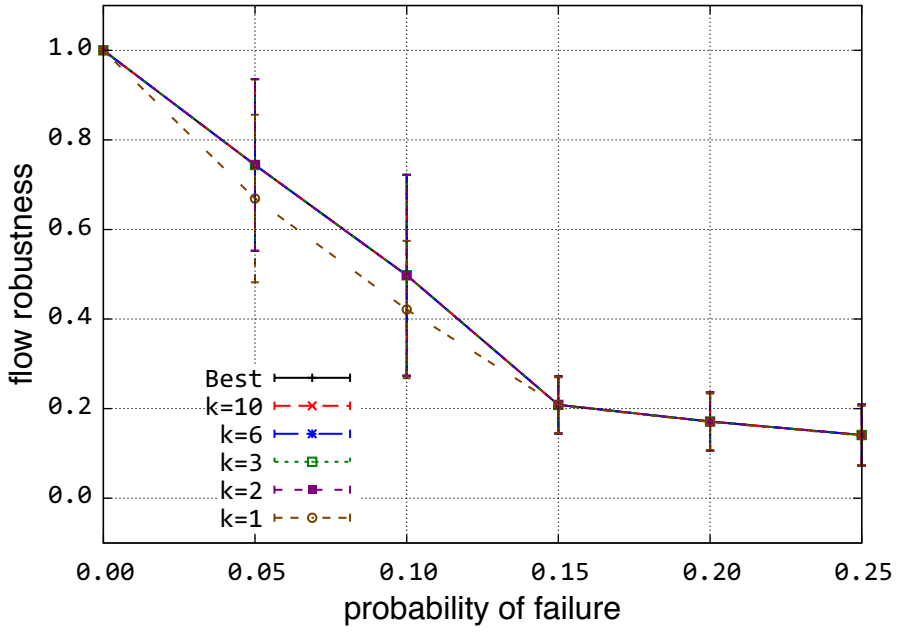

Figure E.188: Flow robustness vs. node failure probability for the Ring topology

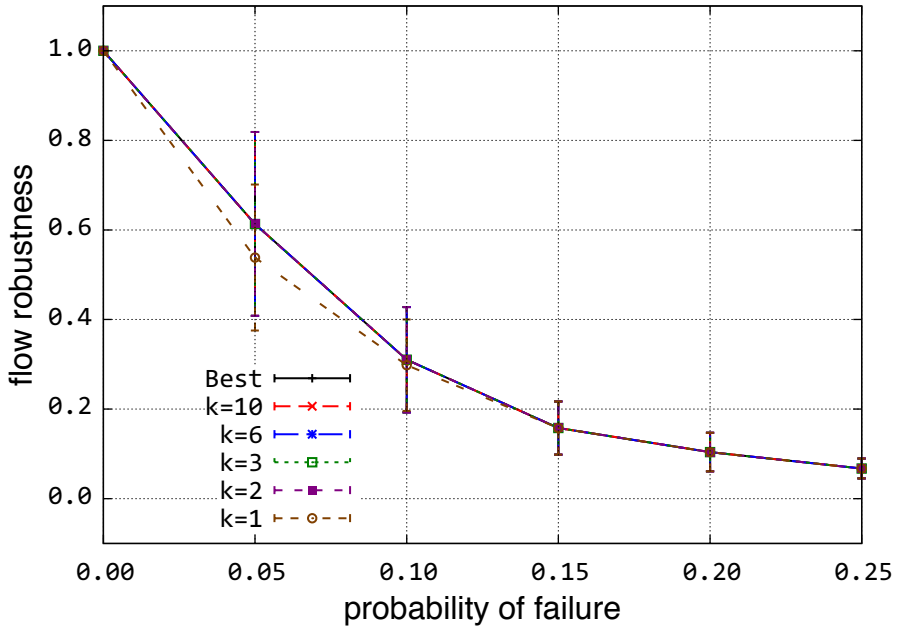

Figure E.189: Flow robustness vs. node & link failure probability for the Ring topology

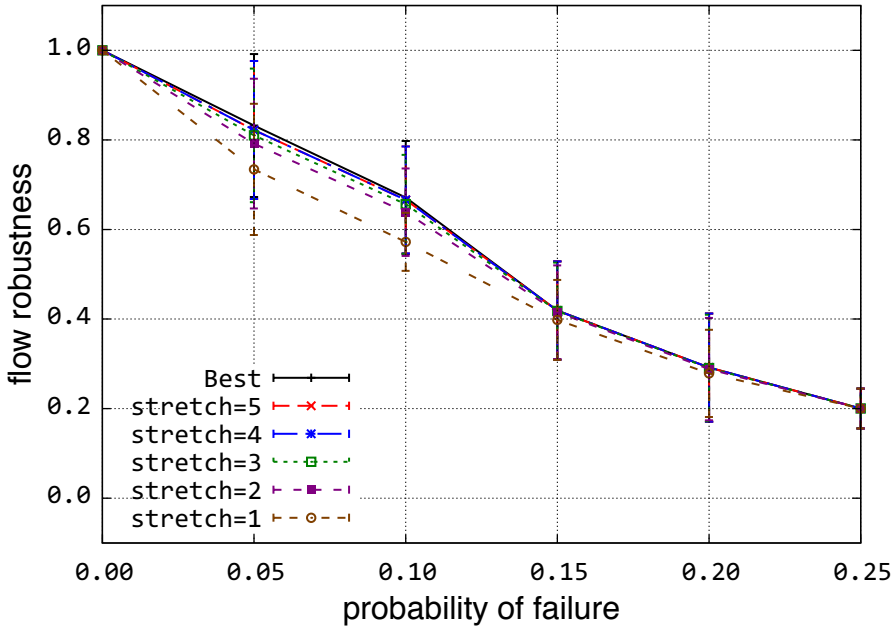

Figure E.190: Flow robustness vs. link failure probability for various stretch limits on the Ring topology

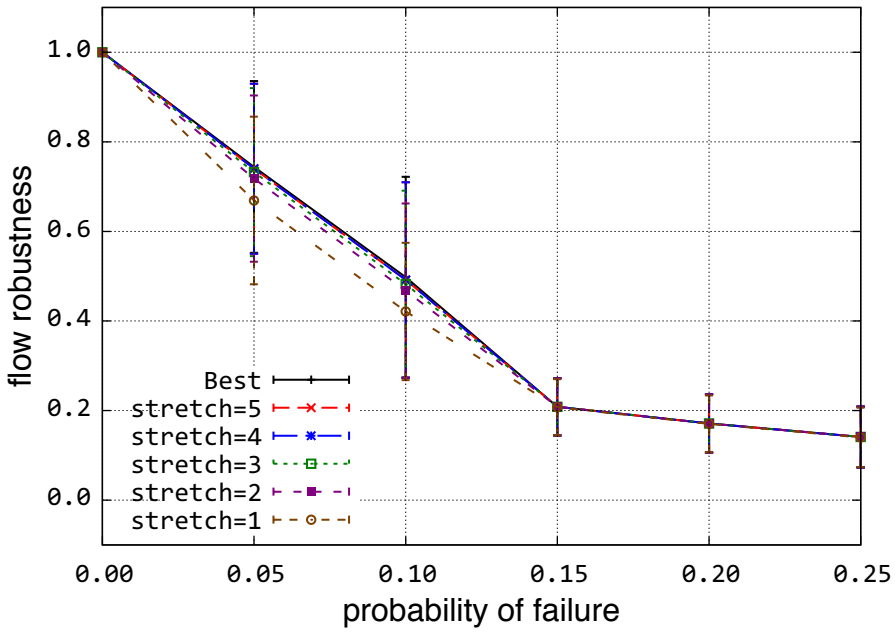

Figure E.191: Flow robustness vs. node failure probability for various stretch limits on the Ring topology

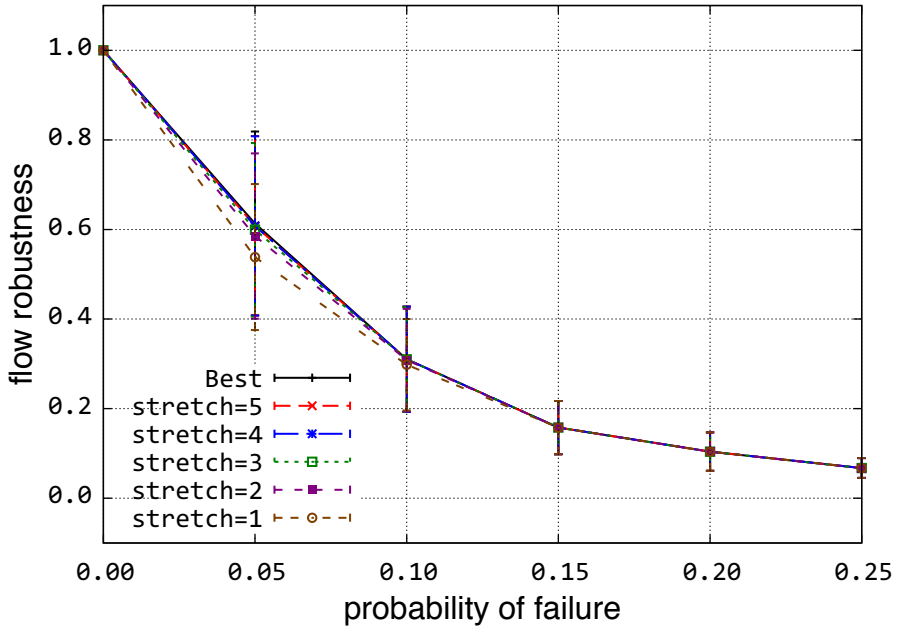

Figure E.192: Flow robustness vs. node & link failure probability for various stretch limits on the Ring topology

E.3.4 Star

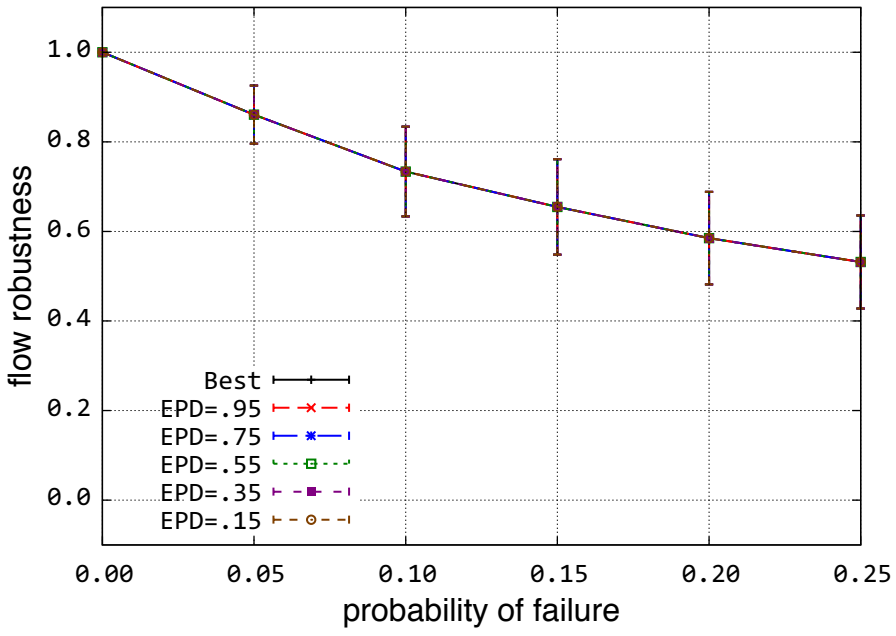

Figure E.193: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 1$ on the Star topology

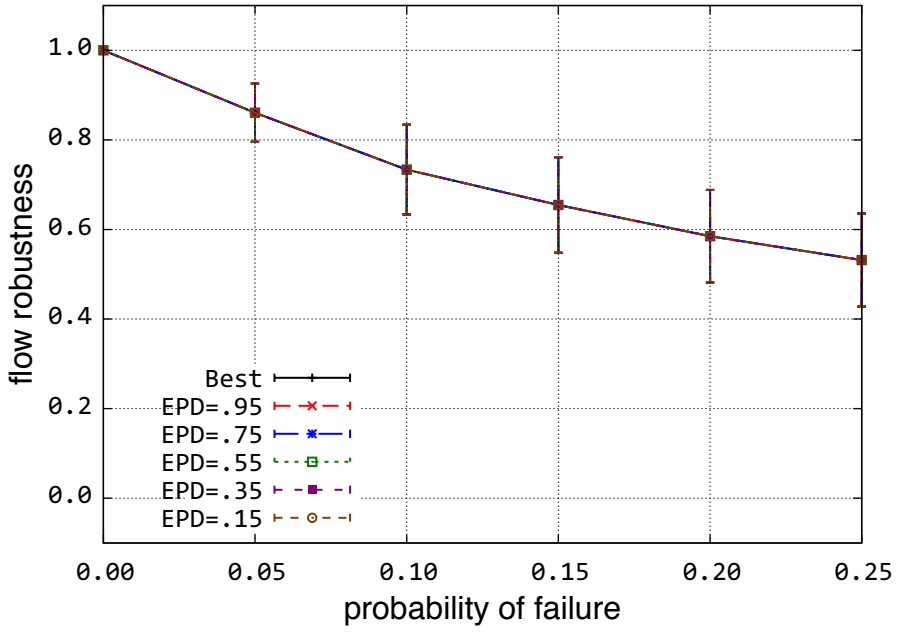

Figure E.194: Flow robustness vs. link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Star topology

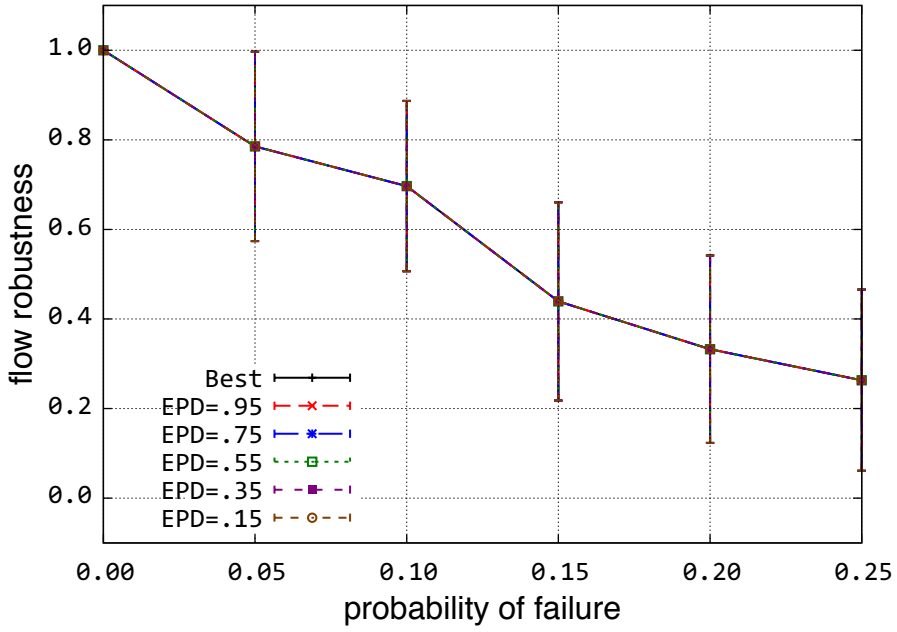

Figure E.195: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 1$ on the Star topology

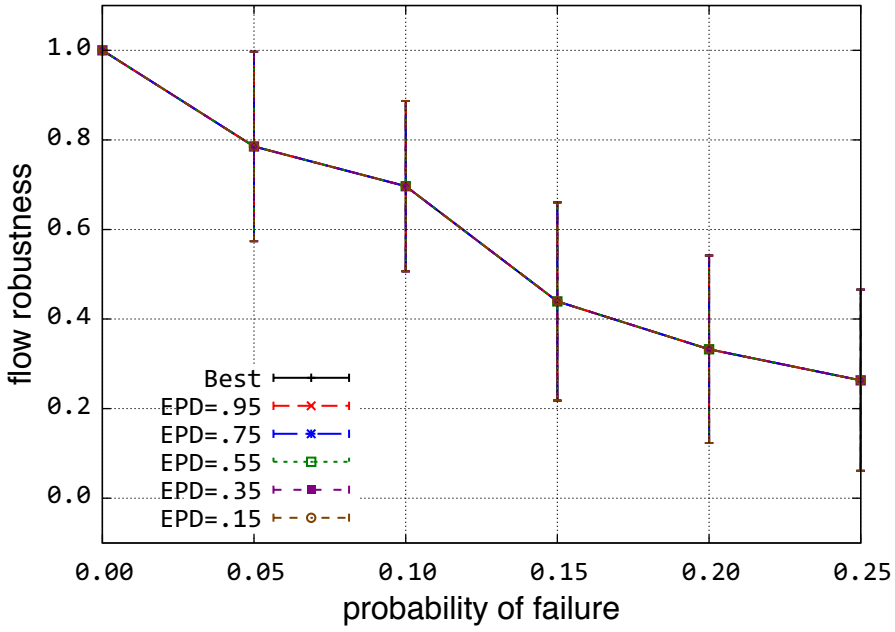

Figure E.196: Flow robustness vs. node failure probability for various EPD thresholds with $\lambda = 0.5$ on the Star topology

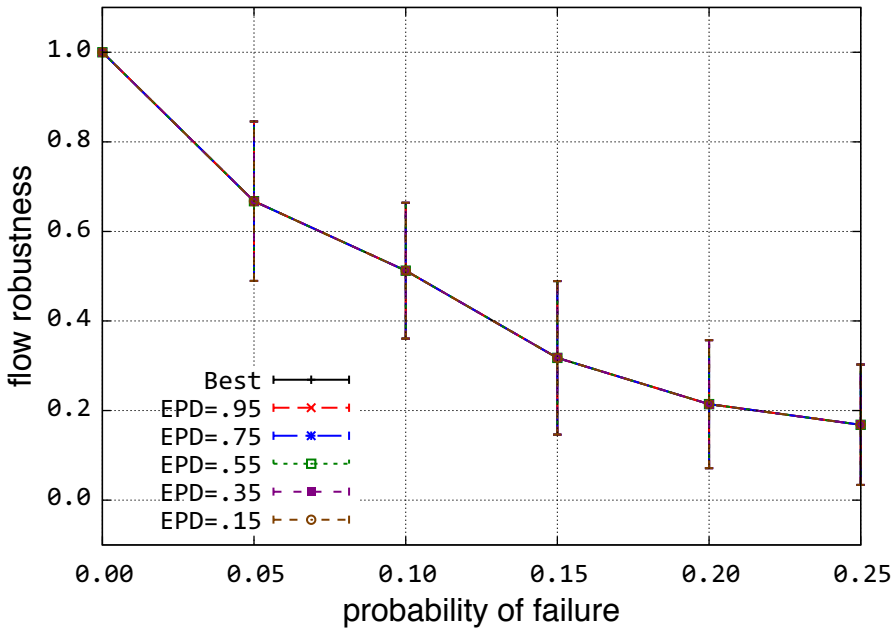

Figure E.197: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 1$ on the Star topology

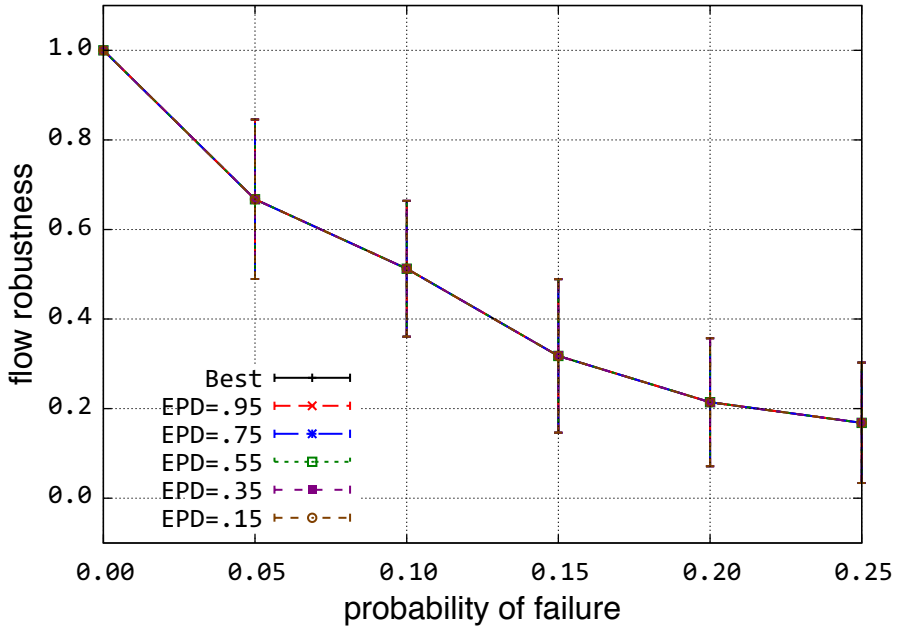

Figure E.198: Flow robustness vs. node & link failure probability for various EPD thresholds with $\lambda = 0.5$ on the Star topology

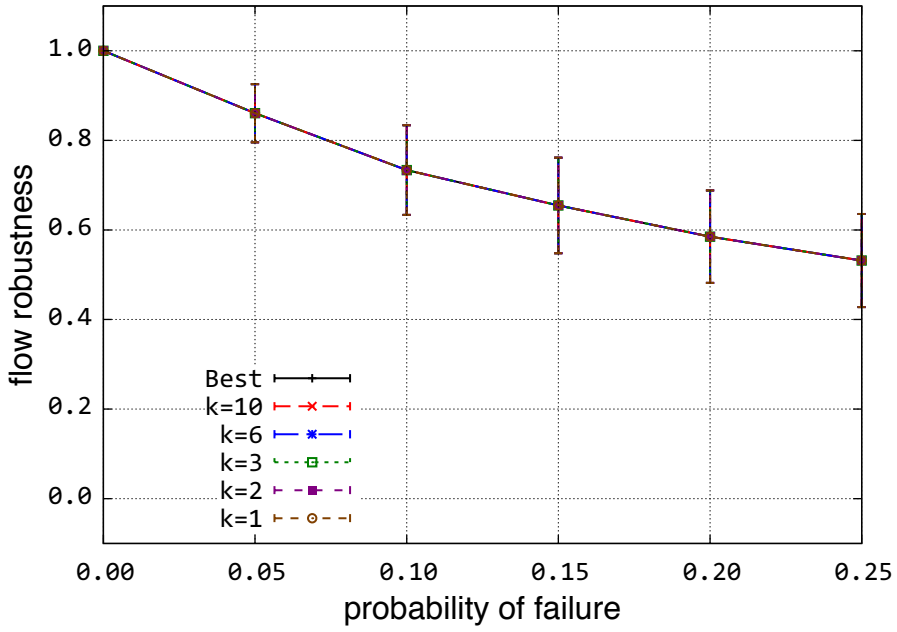

Figure E.199: Flow robustness vs. link failure probability for the Star topology

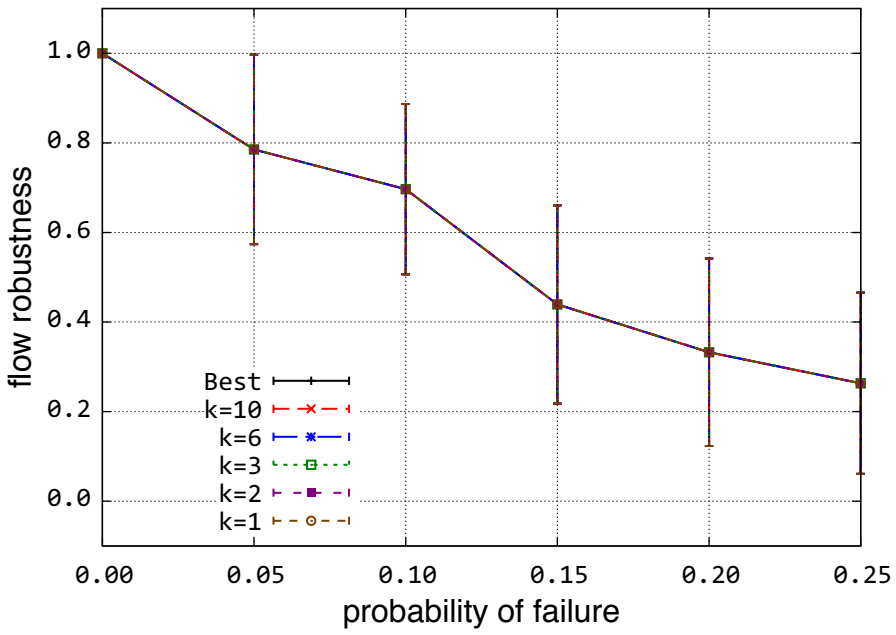

Figure E.200: Flow robustness vs. node failure probability for the Star topology

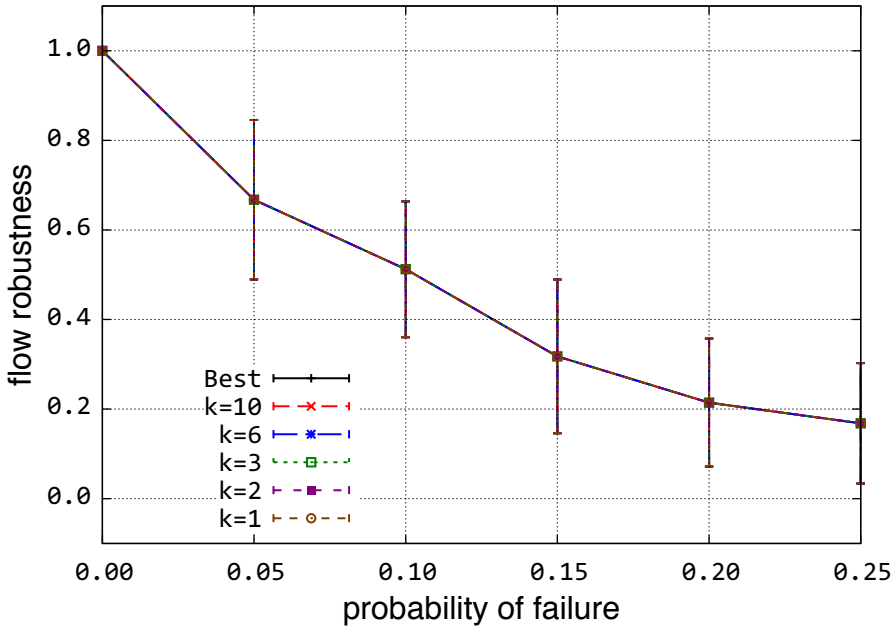

Figure E.201: Flow robustness vs. node & link failure probability for the Star topology

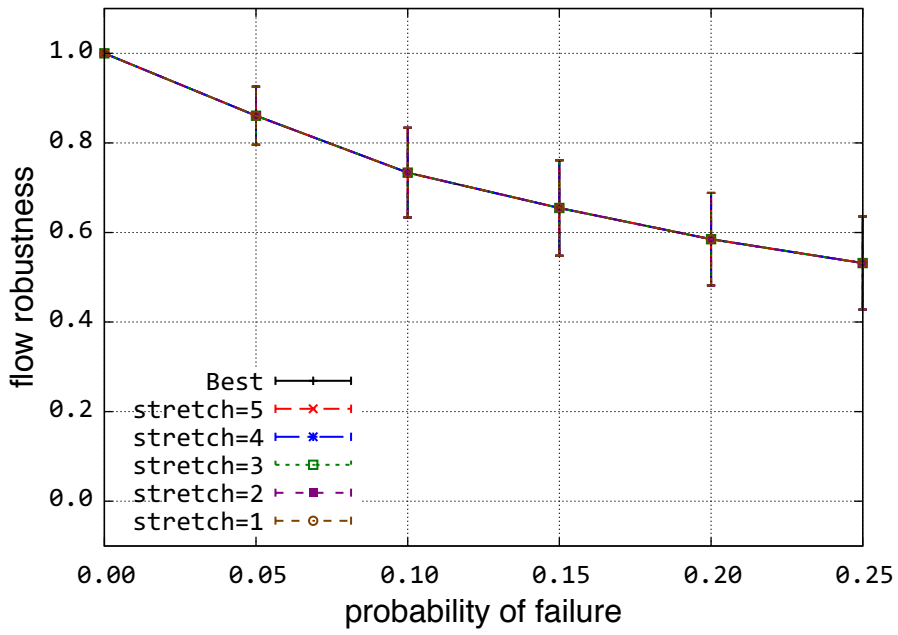

Figure E.202: Flow robustness vs. link failure probability for various stretch limits on the Star topology

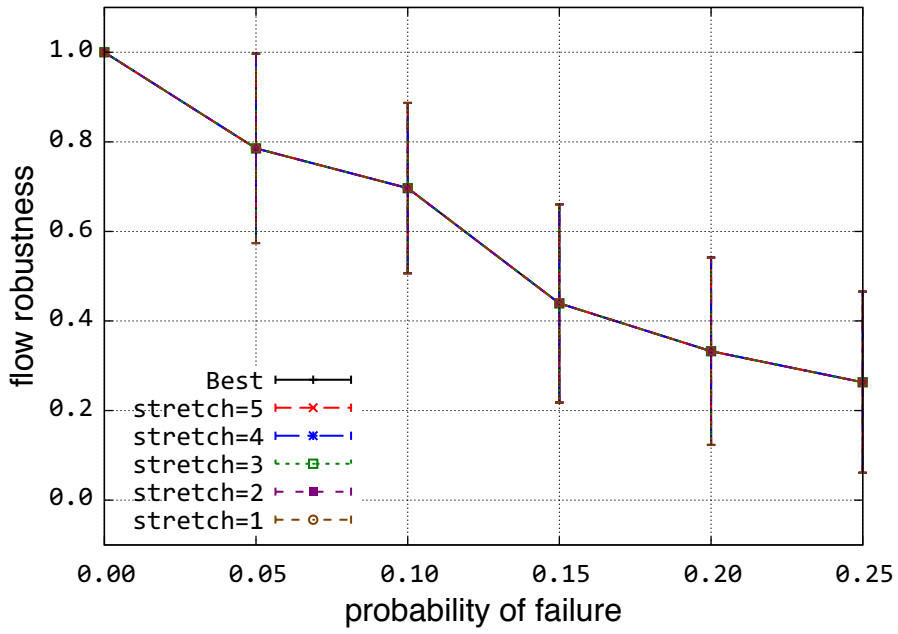

Figure E.203: Flow robustness vs. node failure probability for various stretch limits on the Star topology

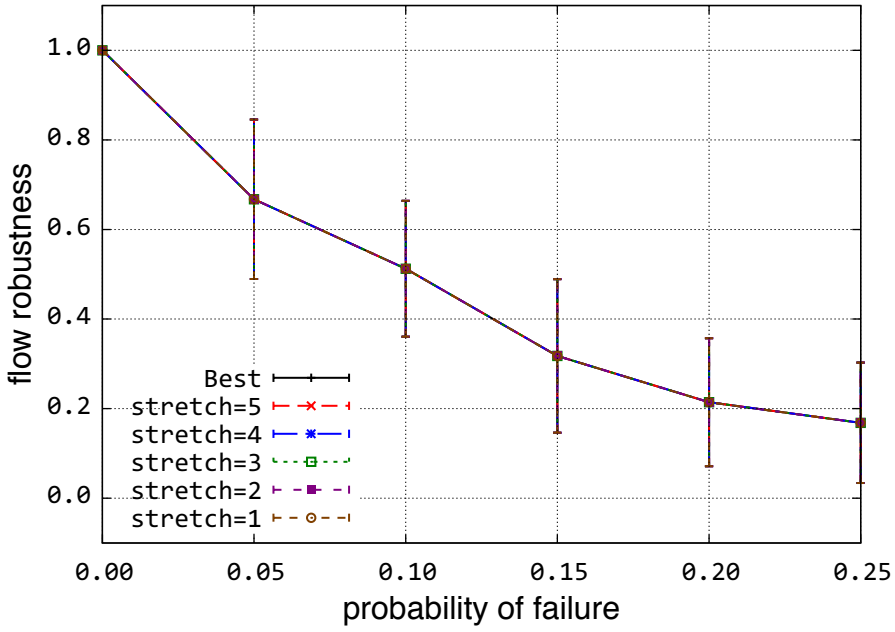

Figure E.204: Flow robustness vs. node & link failure probability for various stretch limits on the Star topology