A. D. MCCXXIV

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

Dottorato di Ricerca in Ingegneria Informatica ed Automatica

Comunità Europea
Fondo Sociale Europeo

RELIABLE EVENT DISSEMINATION FOR TIME-SENSIBLE APPLICATIONS OVER WIDE-AREA NETWORKS

CHRISTIANCARMINE ESPOSITO

Tesi di Dottorato di Ricerca

(XXII Ciclo)

Novembre 2009

Il Tutore

Prof. Stefano Russo

Il Coordinatore del Dottorato

Prof. Luigi P. Cordella

I Co-Tutori

Prof. Douglas C. Schmidt
Prof. Aniruddha Gokhale
(Vanderbilt University)

Dipartimento di Informatica e Sistemistica

🖃 via Claudio, 21- I-80125 Napoli - ☎ *[#39]* (0)81 768 3813 - 🗐 *[#39]* (0)81 768 3816

RELIABLE EVENT DISSEMINATION FOR TIME-SENSIBLE APPLICATIONS OVER WIDE-AREA NETWORKS

By
Christiancarmine Esposito

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
"FEDERICO II" UNIVERSITY OF NAPLES
VIA CLAUDIO 21, 80125 – NAPOLI, ITALY
NOVEMBER 30, 2009

Ποταμοῖσι τοῖσιν αὐτοῖσιν ἔμβαίνουσιν,
ἕτερα καί ἕτερα ὕδατα ἐπιρρεῖ.
Πάντα χωρεῖ καί οὐδέν μένει.
On those stepping into rivers the same,
other and other waters flow.
Everything changes and nothing remains still.
Heraclitus of Ephesus

学而不思则罔
思而不学则殆

Learn without thinking begets ignorance.
Think without learning is dangerous.
Confucius

Table of Contents

Table of Contents iii

List of Tables vi

List of Figures vii

Acknowledgements xi

Introduction 1

1 Data Distribution for Large scale Complex Critical Infrastructures 10

1.1 Large scale Complex Critical Infrastructures (LCCIs) 10

1.1.1 Architectural Evolution of Critical Systems 11

1.1.2 Motivating Example: Air Traffic Management (ATM) systems . . . 18

1.1.3 Data Dissemination Challenges 21

1.2 Publish/Subscribe Systems 24

1.2.1 Generalities 25

1.2.2 Formal Specification 27

1.2.3 Decoupling Properties 29

1.2.4 Notification Architectures 30

1.3 Reliable Publish/Subscribe Services 33

1.3.1 Formal Specification 35

1.3.2 Failure Model 37

1.3.3 Analysis of the state-of-the-art 39

2 Loss Recovery for Reliable Group Communication 45

2.1 Architectures to design a group communication system 45

2.2 Approaches to provide loss-tolerance guarantees 48

2.2.1 Loss Recovery In Transport-Level Multicast 49

2.2.1.1 Centralized Retransmissions 52

2.2.1.2 Hierarchical Retransmissions 57

2.2.1.3 Distributed Retransmissions . . . 58
2.2.1.4 Sender-initiated FEC . . . 59
2.2.1.5 Receiver-initiated FEC . . . 62
2.2.1.6 Hybrid schemes . . . 63
2.2.2 Loss Recovery In Application-Level Multicast . . . 65
2.2.2.1 ARQ Schemes over Overlay Networks . . . 67
2.2.2.2 Lateral Error Recovery (LER) . . . 70
2.2.2.3 Cooperative Error Recovery (CER) . . . 71
2.2.2.4 Gossiping in Overlay Networks . . . 72
2.2.2.5 FEC Approaches over Overlay Networks . . . 74
2.2.2.6 Path Redundancy . . . 76
2.3 Discussion . . . 79
2.3.1 Efficiency of TLM approaches . . . 81
2.3.2 Efficiency of ALM approaches . . . 83
2.3.3 Loss-tolerance of current reliable publish/subscribe services . . . 85

3 Reliable and Timely Data Dissemination over Internet 88
3.1 Two-tier Organization for Data Dissemination Infrastructure (TODAI) . . . 89
3.1.1 Internet Topology . . . 89
3.1.2 Innovative Internet-scale Publish/Subscribe Architecture . . . 91
3.1.3 Building a Multicast Tree . . . 95
3.1.3.1 A DHT overlay: Pastry . . . 95
3.1.3.2 A DHT-enabled Multicast Tree . . . 98
3.1.4 Event Dissemination using the proposed Two-tiers Organization . . 100
3.2 Reliable and Timely Data Dissemination in TODAI . . . 101
3.2.1 Coordinator Replication . . . 104
3.2.2 Reliable and Timely Inter-Cluster Communication . . . 108
3.2.2.1 A Distributed FEC Approach to Tolerate Losses . . . 109
3.2.2.2 Data Dissemination through Multiple Trees . . . 116
3.2.2.3 Integration of Multiple Trees and Distributed FEC . . . 123
3.2.2.4 A reactive technique teamed up with proactive techniques . 126
3.2.3 Reliable and Timely Intra-Cluster Communication . . . 128
3.2.3.1 Layered FEC . . . 128
3.2.3.2 FEC Tuning . . . 130

4 Experimental Evaluation 132
4.1 Preliminaries . . . 132
4.1.1 Selection of the Evaluation Technique . . . 132
4.1.2 Simulator Selection . . . 137
4.2 Experiment Setup . . . 141
4.2.1 Network Topology Generation . . . 142

4.2.1.1 Inter-AS Topology Modeling 144
4.2.1.2 Intra-AS Topology Modeling 147
4.2.2 Network Conditions 149
4.2.3 Failure Modes 154
4.2.4 Workload 156
4.3 Empirical Results 156
4.3.1 Cluster Availability 157
4.3.2 Effective Internet-scale Dissemination 159
4.3.2.1 Distributed FEC 159
4.3.2.2 Multiple trees 162
4.3.2.3 Combination of distributed FEC and multiple trees teamed up with gossiping 166
4.3.3 Effective Cluster-level Dissemination 166
4.4 Outcomes Discussion 167

Conclusions **169**

A Error Correction Coding **171**
A.1 Foundations of coding 171
A.2 Reed-Solomon (RS) Codes 176
A.3 Low-Density Parity-Check (LDPC) Codes 178
A.3.1 Generalities 178
A.3.2 Turbo Codes 180
A.4 Digital Fountain (DF) Codes 181
A.4.1 Luby Transform (LT) Codes 182
A.4.2 Raptor Codes 182
A.5 Discussion 184

B Simulation Environment **187**
B.1 OMNET++ 187
B.1.1 Modular Architecture 188
B.1.2 NED Language 190
B.1.3 Engineering Simulation Programs 190
B.1.4 Simulation Kernel 193
B.2 INET Framework 196
B.3 Rease Topology Generator 197
B.3.1 Topology Generation 198
B.3.2 Traffic Generation 199
B.3.3 Attack Traffic 200
B.4 Oversim 201

Bibliography **203**

List of Tables

1.1 Analysis of the reliable publish/subscribe services available in literature . . 41

2.1 Study of the failures tolerated by the reviewed approaches 85
2.2 Scalability, Reliability and Timeliness analysis of the reviewed approaches . 86

4.1 Criteria for Selecting a Performance Evaluation Technique 133
4.2 Characteristics of the current network simulators 141
4.3 Coverage matrix of the connectivity measures by current topology models . 147
4.4 Parameters and their default values used in the simulations (Time in Seconds) 155
4.5 Parameters and their default values of the workload used in the simulations 156

A.1 Overview of the features of the different coding approaches 184

List of Figures

1.1 Basic scheme of a MCS 12

1.2 Architectures of MCSs: a) point-to-point and b) networked approaches 13

1.3 Architectural overview of a SCADA system 14

1.4 Architectural overview of a generic LCCI 17

1.5 Overview of the European ATM framework 18

1.6 Route tracking of two flights for collision avoidance 19

1.7 Operational schema of a generic event-based architecture 24

1.8 Schematic overview of a generic Publish/subscribe service 28

1.9 Space, time and synchronization decoupling provided by publish/subscribe middleware 29

1.10 Broker network that implements the notification service 31

1.11 A taxonomy of notification service architectures 32

1.12 Efforts on reliability issues in publish/subscribe middleware 35

1.13 Model of the failures that may affect the correct delivery of notifications 37

1.14 Failures treated by reliable publish/subscribe services proposed in the reviewed papers 40

1.15 Layered Organization of a publish/subscribe service 43

2.1 Analysis of the efficiency of the two different perspectives to device Reliable Multicast: a) Transport-Level Multicast and b) Application-Level Multicast 46

2.2 Simple ARQ scheme 50

2.3 Execution of a sender-initiated protocol with a) sender-driven retransmissions and b) receiver-driven retransmissions with receiver-based timeouts 53

2.4 Execution of a receiver-initiated protocol a) without and b) with a logging server . 56

2.5 Tree-based organization of the members within a group: a) overlapping groups and b) separated groups 57

2.6 Execution of gossiping a) in a pure fashion and b) with a logging server 58
2.7 FEC-enabled interactions between a sender and a receiver 60
2.8 Placement of the FEC layer within the stack of protocols for Reliable Multicast: a) layered and b) integrated approach 63
2.9 a) Sender-based, b) Parent-based and Multi-Parent ARQ Schemes 67
2.10 Organization in Tree Planes of LER 71
2.11 Three possible FEC-enhanced methods for loss-tolerant ALM: a) End-to-End FEC, b) Link-by-Link FEC and c) Selective Network-Embedded FEC 74
2.12 Approaches based on path redundancy: a) cross-link, b) in-tree, and c) multi-trees 78
2.13 Classification of all the approaches available in literature to tolerate dropped messages 80
2.14 Analysis of the quality of loss-tolerance for TLM in terms of scalability, reliability and timeliness 82
2.15 Analysis of the quality of loss-tolerance for ALM in terms of scalability, reliability and timeliness 84

3.1 Model of Internet as a composition of Stub and Transit Domains 90
3.2 Event dissemination within an LCCI: (A) within a stub domain and (B) between two distinct stub domains 91
3.3 Layered organization of the data dissemination within an LCCI 92
3.4 Comparison of (A) a pure peer-to-peer and (B) a super-peer topology 94
3.5 Overview of message routing in Pastry 96
3.6 Tree building using Pastry 99
3.7 Event dissemination through the coordinator tree and the clusters 101
3.8 Failures occurring in the Two-tiers Organization of an LCCI 102
3.9 Event dissemination through the coordinator tree and the clusters 105
3.10 Overview of the Selective Network Embedded FEC 109
3.11 Application of the algorithm to elect codes in a multicast tree 114
3.12 Two distinct paths among two nodes placed in two different systems: (A) the paths share a link and (B) the paths are diverse 117
3.13 Innovative joining procedure to build diverse multiple trees 120
3.14 Event delivery among coordinator through multiple trees 122
3.15 Overview of a layered FEC 129

4.1 Testbed used to evaluate the controllability of Internet-scale measurements 136
4.2 Comparison of the variability within the obtained measures 137
4.3 Layered implementation of the simulation model 142
4.4 Topology model composed by two abstract layers 143
4.5 Hierarchical organization of the routers within an AS according the HOT model . 149
4.6 Paths used to characterize the Internet behavior and summary of the obtained characterizations per each path 153
4.7 Trend of A) PLR and B) ABL over all five paths 153
4.8 Variability of ABL during the day on path B and with 200 messages per second 154
4.9 Cluster availability (expressed as number of nines) as function of the number of nodes and the publishing rate 158
4.10 Cluster availability (expressed as number of nines) as function of actively replicated coordinators 159
4.11 Comparison of the performance, in terms of average dissemination latency, between the Scribe implementation of OverSim and the Scribe implementation extended with the Network Embedded-FEC, varying the number of nodes that join a multicast sessions and compose the tree 160
4.12 Dissemination latency at each node of the multicast tree composed of 32 nodes . . 161
4.13 Comparison of the success rate, in terms of ratio of received messages and published ones, between the Scribe implementation of OverSim and the Scribe implementation extended with the Network Embedded-FEC, varying the number of nodes that join a multicast sessions and compose the tree 162
4.14 Trend of standard deviation the dissemination latency at each nodes of a test with 32 nodes joined the multicast session in case of Scribe and NE-FEC Scribe 163
4.15 Results of the simulation study on the multiple tree approach: a) Scalability, b) Timeliness, and c) Reliability 164
4.16 Comparison of performance (a) and reliability (b) varying the applied fault model 165
4.17 Timeliness and reliability of the combination of the proactive techniques 166
4.18 Timeliness and reliability of the combination of the proactive techniques teamed up with the gossiping 167

4.19 Dissemination Latency, timeliness and reliability comparison of the best-effort IP Multicast and Layered FEC 168

A.1 Overview of a communication system 171
A.2 EC problems: A) message corruption and B) message erasure (in figure, the X character indicates the erasure of a digit 173
A.3 Generator Matrix G of an LDPC code and its Tanner graph 178
A.4 Cascaded Graph Sequence that characterize the construction of Turbo Codes . . . 180
A.5 Overview of a raptor code 183
A.6 Comparison among the computational complexity trends 185
A.7 Comparison among the overviewed coding techniques 186

B.1 Layered implementation of the simulation model 189
B.2 Building and running simulation in Omnet++ 191
B.3 Building and running simulation in Omnet++ 193
B.4 Overview of the EthernetInterface provided by the INET framework 196
B.5 Client module of the ARP protocol 197
B.6 Example of AS-level and router-level topology 199
B.7 Architecture of Oversim 201

Volete aver molti in aiuto? Cercate di non averne bisogno.
Do you want to have many people help? Try not need it.

Alessandro Manzoni

Acknowledgements

When a milestone as important as surviving graduate school and achieving PhD is reached, it is natural to spend the first page of a dissertation to thank the countless people who have accompanied me in this long journey of mine for PhD and in one way or another have contributed to my success.

In primis, I thank my family for having tolerated the sudden swings of my mood, which have marked the past three years. Rarely you hear me say something like that, but you were so wonderful with me because all of you were always ready to make me get up when I fell. It is not trivial to tell that I dedicate to y'all this dissertation, without your constant encouragements I would never be able to overcome the various events of deep depression in which I have often felt imprisoned during these three years.

I want to thank all my colleagues and friends at the Mobilab research group: Lelio and Roberto, which have followed the same path as mine and are now getting their PhD along with me, Antonio and Roberto, which have started their journey towards PhD, and Marcello, Generoso and Gabry, which were as our older brothers always available to dispense useful tips. I thank Prof. Stefano Russo, which has always supported my decisions and has been available, despite its commitments, to indicate the best way to achieve my goals, and Prof. Cotroneo, which was able in three years to leave me free to grow and express myself without any interference and has never done groped by the idea of harnessing my creativity.

I'm glad to have spent most of the time in the pleasant environment of the CINI laboratory, attended by wonderful people: Massimo, Antonio, Chiara, Luca, Stefania, Carlo, Peppe, Giulia, Francesca and Nicola, I thank all of you to have cheered up and inspired my days as Ph.D student. I want to extend a special thank to someone really dear to me, which I always carry in my heart: Imma. From the very beginning I've felt you close to me in everyone of my dark moments, and even when our paths have separated, and we lost our daily routine, I've never felt your absence or distance. Thank you for your sincere friendship that helped me so much to pursue my doctoral experience.

I want to thank Prof. Schmidt that made possible my visiting at the Vanderbilt University and allowed me to improve either as a researcher either as a person by broadening my horizons and meeting many persons that marvelously welcomed and treated me. I thank Prof. Gokhale since I am very grateful for all his outstanding help with my research that allowed me to participate in my first important conference as DEBS 2009 and to make up with this dissertation. A special heartfelt thanks I want to turn to Anibal and Sonia. When my family and friends were thousands of miles away and I was facing the experience, fear, of a completely new and different city, country and culture, you have represented my new family that supported me without making me to feel alone, but sharing with me some wonderful moments and experiencing journeys I will never be able to forget. “Chicos, Muchas gracias por darme vuestra amistad. Siempre me he sentido muy cerca, a pesar de la distancia geográfica que nos separa”.

Last, but not least, I want to thank all of my friends:

1. all the guys at the department, Emanuela, Flora, Giuseppe, Carmelo & Carmela, Valentina, Antonio, Vincenzo, with which I have shared so many meals and smiles and organized many great evenings;

2. y'all of “La Finestra Catalana”, Tiziana, Daniela, Nunzia, Virginia, Raffaele, which, with their social events, gave me a way to distract from my thoughts and concerns as PhD student and saved me from an inevitable psychiatric hospitalization;

3. ‘La vecchia guardia”, Imma, Daniele, Mery, Gianni, Peppenzo, Roberto, thanks for be so patient with me and never left me behind despite of all the night spent at studying instead of going out with you.

Napoli, Italy Christian
30th November 2009

Introduction

Context

In the recent decades we have witnessed a massive proliferation of the Internet, which succeeded to pervade all our daily activities and to be adopted throughout the entire world. The emergence of the Internet as a general communication channel is considerably affecting the scale of current software systems and deeply transforming the architecture of future critical systems. Traditionally, a critical system consists of a monolithic, "close world", architecture, i.e., several computing nodes interconnected by a dedicated network with limited or no connectivity towards the outside world. An example of such traditional architecture is Supervisory Control And Data Acquisition (SCADA) [1], *e.g.*, which is used in several current critical systems such as the control room of power plants or air traffic control systems. However, future critical systems will shift to an innovative federated, "open world", architecture, namely *Large scale Complex Critical Infrastructure* (LCCI) [2], which belongs to the group of the so-called Ultra Large Scale (ULS) systems, which were envisioned in a report [3] produced by Carnegie Mellon University's Software Engineering Institute (SEI) in June 2006. Specifically, an LCCI consists in a dynamic Internet-scale hierarchy / constellation of interacting heterogeneous, inconsistent, and changing systems, which cooperate to perform critical functionalities. Many of the ideas behind LCCIs are increasingly "in the air" in several current projects that aim to develop innovative critical systems. For example, EuroCONTROL has funded a project to device the novel European framework for Air

Traffic Management (ATM) in Europe, called Single European Sky (SESAR)[1]. Current European airspace is fragmented in several areas, each one managed by a single control system. Such traditional ATM approach has been demonstrated to be not suitable to handle the future avionic traffic, so it is going to be substituted by a more integrated approach. In fact, SESAR aims to develop a seamless infrastructure that allows control systems to cooperate each other in order to have a wider vision of the airspace, which is no more limited only to their assigned fragment.

As previously stated, traditional critical systems have been characterized by the use of dedicated machines and networks, so hardware and software faults were considered the only threats to the reliability and effectiveness of the system, while communication failures were assumed to be almost improbable to occur. Therefore, in the last decades the research has spent a lot of efforts investigating on how to deal with the former two kind of faults, paying less attention on how to treat communication failures. As a proof of this lack of attention, main standardized, and mature, commercial middleware used in building critical systems do not address them at all, such as Java Message Service (JMS) [4], or provides very basic mechanisms, such as the recent OMG standard called Data Distribution Service (DDS) [5]. However, LCCIs cannot use dedicated networks due to their geographical extension, but they adopt wide-area networks that exhibit an availability between 95 percent and a little over 99 percent [6, 7] and do not provide any guarantees on the offered Quality-of-Service (QoS) [8, 9]. So, when a federated architecture is adopted to device critical systems, communication failures have a high probability to occur, even greater than hardware and software failures, so guaranteeing an efficient data distribution constitutes the pivotal factor to accomplish the mission of LCCIs. The aim of this thesis is to bring a significant contribution in addressing such issue, with the goal of enabling the definition of novel strategies to support effective communication among several critical systems interconnected over wide-area

[1]http://www.eurocontrol.int/sesar/

networks.

Problem Statement

Mostly all the critical systems fall within the wider class of Monitor and Control (M&C) systems [10], *i.e.*, the environment is continuously monitored and the system responds appropriately avoiding threats that may lead to losses of human lives and/or money. For example, an Air Traffic Management (ATM) system keeps track of all the flight in a given portion of the airspace (*i.e.*, the sensing part of the system) and may change the routes of those aircraft that risk to collide (*i.e.*, the responding part of the system). Therefore, one of the main measures to assess the effectiveness of a critical system is timeliness, *i.e.*, a treat has to be detected on time in order to perform proper actions to avoid it. For example, a collision has to be detected within a certain time before its likely occurrence so that aircrafts have time to change their route and prevent the collision to happen. So, critical operations account the right answer delivered too late as the wrong answer, and this means that the adopted middleware has to cope with timing failures and to guarantee that deliveries occur within given deadlines, *i.e.*, <u>on-time information dissemination</u> is required. For example, a radar scans a given area of the airspace hundred times in a second, and a control system usually combines the data received by several radars to view the position of all the aircrafts in a given portion of the airspace. If a message produced by a radar reaches an ACC later than 0,6 seconds, it is not usable since the current state of the flights does not match the content of the received message, and the control system that receives it has an out-of-time view of the position of the aircrafts. This can cause disastrous consequences: when late-received radar data are fused with the timely-delivered ones, several false positives and false negatives can be generated through the process of collision detection

As previously asserted, message deliveries over wide-area networks exhibit not-negligible bursty loss patterns, *i.e.*, a message has a considerable probability P to be lost during the delivery and the succession of consecutive dropped messages has an average length ABL greater than two. The critical nature of LCCIs demands that messages have to be delivered to all the destinations despite of the faulty behaviour of the network, so the adopted middleware has to provide some means to tolerate the message losses imposed by the network in order to achieve a reliable message distribution. However, the reliability gain is always achieved at the expenses of worsening the predictability of the delivery time and leading to timing failures. Since LCCIs require that messages are guaranteed to be timely delivered to all the interested consumers despite of the occurrence of several failures, it's needed to provide a trade-off between the achievable reliability and timeliness degree.

The ultra large scale of LCCIs worsens the already-tough challenge to join reliability and timeliness since several solutions to tolerate message drops exhibit severe scalability limitations [11]. In addiction, since LCCIs are smeared on several networking domains due to their geographic distribution, network conditions, *i.e.* propagation latency and loss pattern, do not result uniform all over the infrastructure, but the overall LCCI is composed of several portions each one characterized by a particular configuration of the network behaviour. Therefore, the approach "one solution fits all" does not work in the case of LCCIs, but the adopted middleware has to autonomously choose the proper message delivery strategy to the experienced network conditions in order to support a reliable and timely data distribution. Last, wide-area networks do not exhibit a stable behaviour but network conditions continuously change. Therefore, the adopted delivery strategy has also to provide self-configuring capabilities in order to adapt to any changes in the behaviour of the underlying network and to provide almost the same reliability and timeliness degree masking fluctuations in the network conditions.

Open Issues

Publish/Subscribe interaction model [12] is an asynchronous messaging paradigm where consumers, namely subscribers, receive only the messages produced by the so-called publishers in which they have expressed interest through a subscription predicate. Middleware services based on such model are suitable to be adopted in ULS systems since they exhibit strong decoupling properties that enforce scalability. As previously mentioned, LCCI require a reliable data distribution, so the adopted publish/subscribe service needs to provide means to cope with message losses. However, since its inception, the publish/subscribe community has been more focused on scalable architectures, efficient delivery, and expressive subscriptions rather than reliable event dissemination. However, this status quo is changing as more and more publish/subscribe services have started to be used in application domains that expose stringent reliability requirements. For example, a middleware complaint to the recent specification standardized by OMG for publish/subscribe services, namely Data Distribution Service (DDS) [5], has been used in the novel combat management system proposed by Thales, namely TACTICOS [13], to manage all the C4I functionalities on several kind of warships.

Most of the research efforts on investigating reliable publish/subscribe services focused on how to maintain the connectivity among publishers and subscribers after the occurrence of failures [14, 15], while less interest has been put on dealing with message losses [16, 17]. This is due to consideration that a publisher/subscriber service can be built on top of a given multicast protocol, and how to achieve loss-tolerance has not been felt challenging since it can be resolved by using a reliable multicast protocol. However, the loss-tolerance issues raised by LCCIs are far from been completely treated even using one of the best protocols available in the literature of the reliable multicast. In fact, most of the current reliable multicast approaches adopt *reactive techniques* to tolerate message losses, *i.e.*, a dropped message is somehow detected by one of the destinations and a retransmission is triggered so that the message can be recovered. Such techniques allows guaranteeing an high degree of loss-tolerance, but there are no assurances on the timeliness of the deliveries. In fact, the number of retransmissions needed to successfully deliver a message depends on the number of consecutive messages dropped by the network. However, since ABL is not known a priori, it is unlikely to forecast how many retransmissions are needed to deliver a message, so the time to deliver a message in case of drops is not predictable and timeliness is not achieved. On the other hand, *proactive techniques*, *i.e.*, necessary countermeasures to deal with possible message drops are taken prior their happening, are the only feasible means to guarantee both timeliness and reliability. In fact, such approaches minimize the time to recover a lost message, so communications over faulty networks do not suffer of performance fluctuations. However, current proactive approaches exhibit scalability and reliability limitations that prevent their usability in the context of LCCI. In addiction, current reliable publish/subscribe services that treat message losses suffer of two main drawbacks: (*i*) the event dissemination strategy is not chosen with respect to the specific conditions experienced by the network and (*ii*) the same strategy is adopted all over the service even if it is segmented in portion with different network conditions.

Thesis Contribution

The aim of this PhD thesis is to bring a significant contribution in the area of reliable multicast, with the goal of enabling the definition of novel strategies to provide both reliability and timeliness in large scale critical systems. The efforts striven in this dissertation result into the design of *(i)* innovative proactive techniques for Application-Layer Multicast (ALM) and *(ii)* a hierarchical cluster-based overlay network that optimizes the adopted reliability strategy according to the behaviour of the given routing domain.

During the three years experience of the doctorate the main proactive methods have been investigated:

Forward Error Correction [18] : additional data is coded from the message, so that the destination can recover the lost packages by decoding the received packages. FEC techniques exhibit the drawback that the coding actions are focused on the message senders leading to evident scalability problems. In fact, the redundancy degree is tailored on the destination that experiences the worst loss patter leading to unneeded traffic towards destinations that experience better loss patter than the worst one. This dissertation overcomes such issue by proposing a <u>decentralized FEC technique</u> where only a sub-set of interior nodes of a multicast tree performs coding actions. This allows to distributed the coding duties in several parts of the dissemination infrastructure and experimental data proved that scalability is proved respect to traditional FEC without affecting the achievable reliability degree.

Multi-tree Dissemination [19] : studies, such as [20], have demonstrated that topology of the Internet is characterized by an intrinsic path redundancy degree, *i.e.*, there are several distinct paths from a source to a given destination. Multi-tree dissemination takes advantage of such redundancy by sending several copies of a message through several paths towards the given destination. Reliability is achieved if the multicast forest, *i.e.*, the several multicast trees built by the system among the subscribed applications, verifies the essential requirement of *Path Diversity* [21], the several paths connecting a source to a destination do not have to contain overlapping network devices. In fact, if path diversity is not verify, it is possible that the failure of an overlapping devices can leave to loss of all the messages and a destination can experience message drops. This dissertation resolves such matter by proposing a novel algorithm to built diverse trees.

The last contribution of this thesis is a hierarchical architecture of a event dissemination solution for LCCIs. Such architecture is an hybrid peer-to-peer topology, characterized by two different layers: *1)* the low layer is composed by a pure peer-to-peer cluster, and *2)* the higher layer consists of a network of all the coordinators of each cluster of the LCCI. The cluster can be defined at deployment time of each system of the LCCI (e.g., each cluster consists of an entire system), or at run time, using a given proximity measure in order to group all the closer peers (e.g., each system may be made of several clusters). On one hand, each system administration can choose a reliability strategy tailored on the routing domain of the managed system, without considering the choice in other systems. In fact, to assist in such choice, this thesis presents a study of how the different available reliability means for ALMs perform under different network conditions. On the other hand, the network of the coordinators uses a single reliability technique that is able to guarantee event delivery even in the worst loss pattern. Since communications among coordinators have to be as much reliable and timely as possible, only the previously introduced proactive techniques are used.

Thesis Organization

- Chapter 1 provides the basic concepts of LCCIs and a description of the LCCI example, which motivates the studies which have been conducted in this thesis, concluding by emphasizing the problem of data dissemination in the context of LCCIs. On the other hand, it also gives an overview of publish/subscribe middleware, focusing on the issues of supporting reliable event dissemination. In addition, it describes the related literature on reliable publish/subscribe services, highlighting their limitations with respect to the requirements imposed by LCCIs.

- Chapter 2 provides an overview of the loss-tolerance approaches available in literature and analyzes their pros and cons.

- Chapter 3 describes the proposed hybrid peer-to-peer topology to architect a data dissemination service suitable for LCCIs and the reliability strategies that are needed to assure a timely and reliable data dissemination among the systems composing an LCCI.

- Chapter 4 is finally devoted to the experimental campaign performed to assess the effectiveness of the approaches proposed in this dissertation.

This dissertation contains also two appendixes that are focused on coding techniques to implement a Forward Error Correction strategy (appendix A) and a description of the simulation environment used to performed the experiments whose outcomes are described in chapter 4 (appendix B).

Chapter 1

Data Distribution for Large scale Complex Critical Infrastructures

Innovative critical systems that are currently being developed or are planned to be realized in the near future are going to adopt federated architectures, which strongly affect their scale, complexity and communication requirements. Their demand for scalability in the information dissemination is leading to a growing interest in publish/subscribe middleware to glue together the several systems forming these federated architectures. This chapter examines the causes and consequences of rapidly increasing scale in the critical systems of the future and introduces the notion of publish/subscribe service. Then it reviews the recent experience on reliable publish/subscribe services.

1.1 Large scale Complex Critical Infrastructures (LCCIs)

We are witnessing a radical evolution in the architectures adopted to design critical systems, caused to optimize the control efforts in critical assets such as transport management, power grids or financial services. In fact, Large-scale Critical Infrastructures (LCCIs) are progressively imposing versus the traditional "isolated" architectures as the novel paradigm for designing future critical systems. This section aims at address the following objectives:

1. describing in subsection 1.1.1 how and why the architectures adopted to build critical systems have evolved over the years starting from dedicated point-to-point connections passing by SCADA systems and arriving to the innovative LCCIs;

2. illustrating in subsection 1.1.2 a practical example of LCCIs taken from the application domain of the Air Traffic Management[1];

3. discussing in subsection 1.1.3 the novel requirements imposed by LCCIs on the adopted data dissemination infrastructure and the limitations of current architectures.

1.1.1 Architectural Evolution of Critical Systems

Monitor and Control Systems (MCSs) [23] are special-purpose engineering artifacts designed to automatically sense the pulse of a given system and defend it against exogenous events by adjusting its execution parameters. MCSs are tightly intertwined in our daily lives so much that we take them for granted. For instance, there are more than a handful of computer control systems in a typical car that we now drive. In everything from the engine to transmission, shock absorber, brakes, pollutant emission, temperature, and so forth, there is an embedded microprocessor controller keeping an eye out for us. Under the umbrella of MCSs, there exists a particular class of systems that are employed to control critical processes, such as nuclear reactors, avionic traffic or electric generation, transmission and distribution. Such systems are defined *critical* since their failure can cause losses in terms of human lives (in this case they are referred as *safety-critical systems*) and/or money (in this case we have *mission-* or *business-critical systems*) [24].

MCSs are typically composed of three components (as shown in Figure 1.1): i) *Sensors*,

[1]The work described in this dissertation has been conducted within the context of an industrial collaboration, called Software Initiative (www.iniziativasoftware.it), between the Computer Engineering Department (DIS) of the University of Naples Federico II and the Selex Integrated Systems of the Finmeccanica group. Selex is a leading Italian company in designing large-scale systems for Homeland Protection, Air Defence, Battlefield Management, Naval Warfare and Air Traffic Control, and provided the representative example of characteristics and requirements of LCCIs used within the research efforts described in this dissertation.

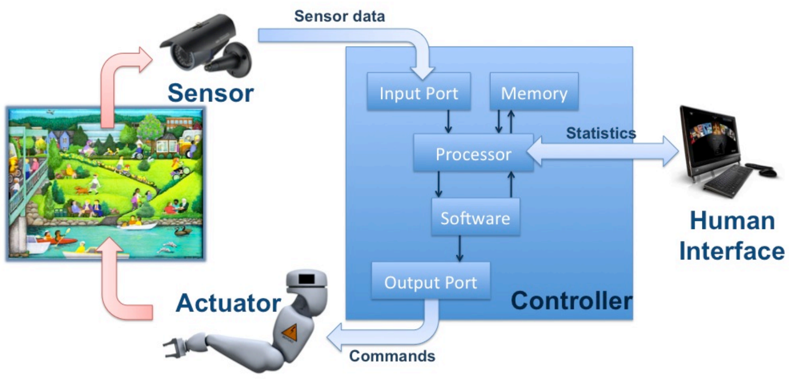


Figure 1.1: Basic scheme of a MCS

which monitor the behavior of the system under control, ii) a *Controller*, which returns the commands obtained by the execution of some specific control algorithms based on the data received by the sensors, and iii) *Actuators*, which transform the commands received by the controller in proper control actions performed on the system under control. Optionally, there may be a fourth component, namely a *Human Interface*, that collect statistics from the controller to provide a human operator with a view of the state and behavior of the system under control. A MCS is characterized by intense information flows among its components, and the achievement of the mission of the system depends on how effectively data is distributed. Although MCSs have been used in several different domains, spanning from military applications (*e.g.*, defending ships against missile attacks or controlling unmanned combat air vehicles through wireless links) to civil applications (*e.g.*, regulating the temperature of coolant in nuclear reactors and maintaining the safe operation of steel manufacturing machinery), all these heterogeneous use cases have one feature in common: the right answer delivered too late becomes the wrong answer [25]. Since the reactivity of embedded systems depends on the time to exchange information among its components, it

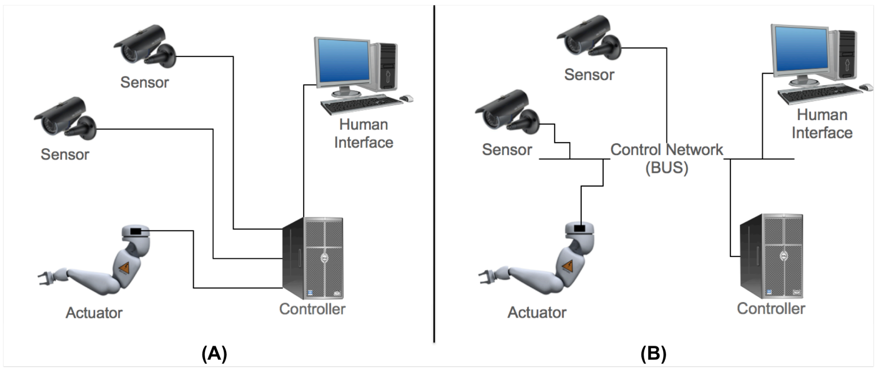


Figure 1.2: Architectures of MCSs: a) point-to-point and b) networked approaches

is crucial that the adopted communication infrastructure guarantees a timely delivery of data produced by the several elements in a MCS. For example, if the monitoring data sent by the sensor do not reach the controller on time, or even worse they are lost, the MCS is not able to timely perform appropriate actions to treat unexpected incorrect behavior of the system under control, leading to damages to the system or disastrous consequences in case of critical MCSs.

The typical interconnection topologies for MCS, which have been successfully adopted for decades, are characterized by point-to-point connections among sensors, controller and actuators [26], as shown in Figure 1.2a: dedicated wires directly interconnect sensors, actuators and an optional human interface to the controller, which plays the role of the center of a star topology. However, such a centralized point-to-point approach exhibits severe scalability, reliability and maintainability limits and has been proved to be not suitable to achieve novel requirements, such as modularity, decentralized control, and low deployment costs. Therefore, in the 1990s, we witnessed a shift towards the new approach [27]

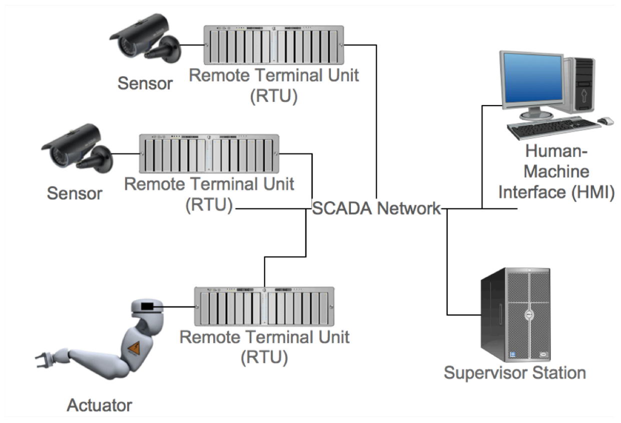

Sensor
Remote Terminal Unit (RTU)
Sensor
Remote Terminal Unit (RTU)
SCADA Network
Human-Machine Interface (HMI)
Remote Terminal Unit (RTU)
Actuator
Supervisor Station

Figure 1.3: Architectural overview of a SCADA system

illustrated in Figure 1.2b, which gave the birth of a novel generation of MCSs called **Networked Control Systems** (NCSs) [28]. The components of NCSs are no more connected through dedicated point-to-point connections but common-bus network architectures, such as Ethernet or a Local Area Network (LAN), are introduced. The adoption of this networked approach has made it possible to improve efficiency, flexibility and reliability and to reduce installation, reconfiguration and maintenance efforts [26]. NCSs are usually implemented as **Supervisory Control and Data Acquisition** (SCADA) systems [1], so that over time the two terms have become synonymous. In fact, SCADA systems are composed of a *Human-Machine Interface* (HMI), which allows a human operator to monitor and control a process; a *Supervisor Station*[2], which gathers data on the process under control and sends commands; and several *Remote Terminal Units* (RTUs), which connect to the physical equipment by converting electrical signals from the equipment to digital values for

[2]The increase of the complexity of current NCSs has made unsuitable to concentrate in a central component the task of gathering all sensor data and processing them in order to maintain a process within a desired behavior [29]. In fact, nowadays SCADA systems are also used to indicate the so-called **Distributed Control Systems** (DCSs) [30], *i.e.*, systems that distribute control in multiple devices rather than a single centralized one that control all the system.

the supervisor station or vice versa. In the first case, the RTU plays the role of a sensor by feeding the supervisor station with data about the current state of the process under control, while in the second case, it works as an actuator by converting and sending out commands to the equipment. Comparing in Figures 1.2 and 1.3, it can be seen that there is a direct mapping between the components of a generic NCS and of a SCADA system.

Due to security and availability concerns, NCS/SCADA solutions, especially when they represent critical systems, traditionally adopt architectures based on a monolithic, "close world", perspective, *i.e.*, several computing nodes interconnected by a dedicated network with limited or no connectivity to the outside world. This brought critical infrastructures, *e.g.*, telecommunications, water supply, electric grids or transportation systems, being fragmented in several MCSs, each one focused on controlling only one portion of the overall infrastructure and blind on what is happening in other portions due to the lack of cooperation and coordination with the other MCSs. Let consider the case of an *Power Grid* (PG), which covers all the necessary devices to transport electric power from production plants to final users and control systems to maintain the correct balance between load and production. Specifically, the current European PG is segmented in several national PGs, each one managed by a certain organization called *Grid Manager* (GM). Unfortunately, there is no orchestration among the different GMs, only sporadic data exchanges using phone calls between human operators. Despite such fragmentation, there are considerable interdependencies among the different national PGs[3] and this implies that treats to a given PG may also be caused by faults that occur in an other PG, so a GM needs to have the view of what

[3]For example, less than 10% of the Italian electricity demands during the daylight and picks of 25% during the night are covered by electricity imported from neighboring countries such as France and Switzerland [31].

is happening in other PGs in order to timely react to these propagated faults. A practical example that highlights the need of cooperation among control systems in current critical infrastructures is the huge blackout that happened in Italy on 28th September 2003. Such massive power failure, *i.e.*, twelve hours without electricity, was not caused by an internal fault within the italian power grid, but by the interruption of the high voltage transmission line that brings electricity from Switzerland to Italy. This interruption was communicated to GRTN, *i.e.*, the Italian GM, via a phone call to a human operator, however due to misunderstandings no countermeasures were taken. On the contrary, if the control equipment of the italian PG was fed directly with monitoring data of the swiss PG, the sudden drop in electricity supply would be automatically addressed without any corrective action introduced by a human operator.

Recent developments in networking and the extraordinary success of Internet as communication channel are making possible the gradual abandon of this traditional, ineffective, perspective and encouraging an architectural revolution that is leading to a new, third, generation of MCSs, the so-called **Large scale Complex Critical Infrastructures** (LCCIs) [2], which represent an example of the *Ultra Large Scale* (ULS) systems that has been envisioned in a report [3] produced by Carnegie Mellon University's Software Engineering Institute (SEI) in June 2006. As shown in Figure 1.4, LCCIs adopt a federated, "open world", perspective, *i.e.*, they consist of a dynamic Internet-scale hierarchy/constellation of interacting heterogeneous, inconsistent, mostly already-existent, legacy systems, which cooperate to perform critical functionalities. LCCIs provide a solution to the unsuitability

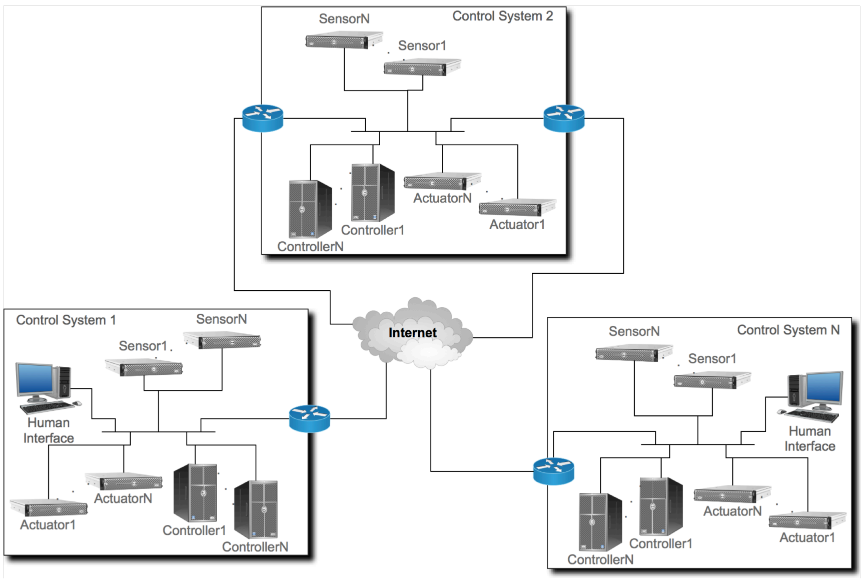


Figure 1.4: Architectural overview of a generic LCCI

of traditional architectures, and to the urgent need for a more integrated control architecture. In fact, the federation that characterizes LCCIs goes beyond the trivial monitoring data exchanges between distinct control systems, but allows implementing complex decentralized and distributed control algorithms where decisions are taken by the cooperation of several controllers geographically distributed. Such collaborative intelligence underlying the control decision-making process is necessary since control decisions taken in a given portion of a critical infrastructure may affects the other portions, as noticed in the collapse event described previously. Many of the ideas behind LCCIs are increasingly “in the air” in several current projects that aim to develop innovative critical systems. The next subsection describe an on-going project carried by EuroControl, i.e., the European Organization for the Safety of Air Navigation, to devise the pan-European air traffic management, which provides a representative example of LCCIs, useful to point out the characteristics and

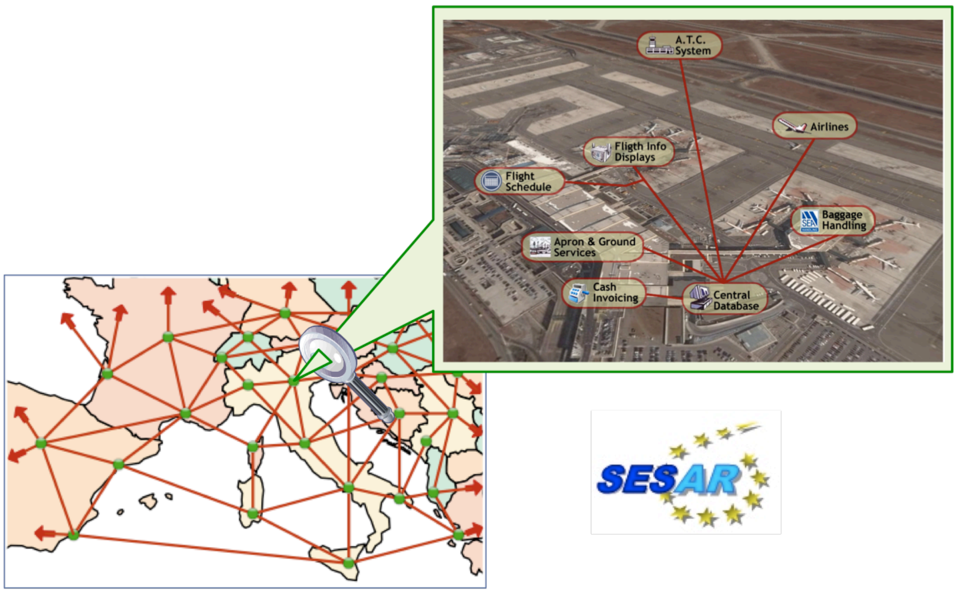


Figure 1.5: Overview of the European ATM framework

requirements of such infrastructures.

1.1.2 Motivating Example: Air Traffic Management (ATM) systems

Air Traffic Management (ATM) refers to a set of complex control services that consists of ground-based controllers who control aircraft movements on the ground and in the air. As in other similar infrastructures, the current approach to design an ATM framework is to partition it into several systems, called *Area Control Centers* (ACCs), as illustrated in Figure 1.5. Each ACC is composed of hundreds of interconnected devices responsible of performing all basic ATM functionalities, among which the most critical one is en-route and landing/departing control of aircrafts in a given portion of the airspace [32], named *Area of Responsibility*(AoR). However, control elements in a given ACC do not directly exchange information with related elements in other neighboring ACCs, but the control of a flight is handed over from an ACC to another one only via radio communications

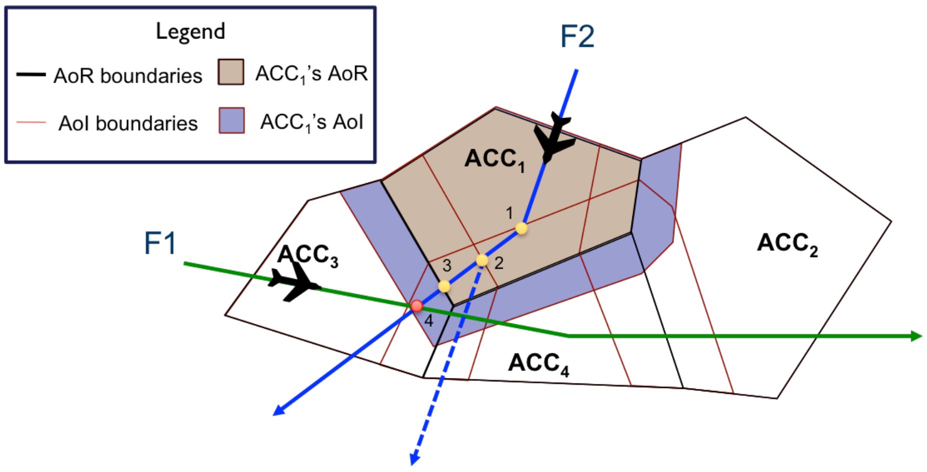


Figure 1.6: Route tracking of two flights for collision avoidance

among human operators, similarly to what we have seen in power grids. This mechanism to hand-over the control of a flight among different ACCs has been proved to be unsuitable to handle the growing avionic traffic in Europe and may cause severe effects. Let consider Figure 1.6 that shows the route of two flights, namely F1 and F2, that pass through four ACCs. The aircraft F2 is currently flying in the airspace assigned to ACC$_1$, while F2 is flying in the airspace of ACC$_3$. When there's no cooperation, each ACC does not possess any information on flights in volumes of the airspace different that its own. Specifically, ACC$_3$ is unaware that F2 represents a serious treat to aircraft F1. In fact, if neither one of the two aircrafts will change their route, a collision will occur in point 4 of Figure 1.6. Since ACC$_1$ does not have any data concerning F1 since it is completely out of its AoR, it cannot command any change of route to the pilot of F2. On the other hand, ACC$_3$ will achieve from its radars the position of aircraft F2 only when it will enter it AoR by passing over point 3, maybe too late for the pilot of F1 to take the needed countermeasures to avoid the collision. When ACC_1 is approaching to handover the control of F2 to ACC_3, its operator contacts the other ACC to inform of the route taken by the aircraft. However, such information cannot be timely delivered since exchanged among two human operators using radio communications. In addition, for some unexpected reasons, *e.g.*, turbulence, the actual position of the aircraft can diverge, referring to the dashed blue line, and the control needs to be performed as fast as possible, maybe handing the control over the flight to another ACC, in the example ACC_4, different that the foreseen one, as ACC_3.

The growing of the avionic traffic is progressively reducing the distance between the flights, making the airspace more crowded. Fast control handover is crucial to avoid dangers to the human lives in the European skies, so a more computer-based solution for the control handover is needed. Specifically, each ACC will be characterized by an Area of Interest (AoI), overlapped with the AoR of other ACCs, as illustrated in Figure 1.6. This way, when F2 enters the AoI of ACC_4 by surpassing point 1, radar data with the flight information of F2 will be delivered to the control systems of ACC_4, so it can be aware that an aircraft is close of its AoR and know its current position. When F2 reaches point 2, it enters AoI of ACC_3, and flight information are sent also to it. This way both ACC_3 and ACC_4 have an updated view of the position of F2: ACC_3 can preemptively change the route of F1 or ACC_4 can immediately notice a change in the route of F2 and consequently take proper actions. In addition, control decisions are not taken only by each ACC for the flights in its fragment of airspace but such decision making process is perform in a cooperative manner among neighboring ACCs. Since this solution allows improving the quality of the flight management, EuroCONTROL has taken it moves to devise the novel cooperative

European ATM framework by funding a project called Single European Sky (SESAR) [33]. Such project aims to develop a seamless infrastructure to allow control systems to cooperate in order to have a wider vision of the airspace and better handle the growing avionic traffic in Europe. The key idea is to adopt a middleware solution that glues together all the European ACCs by conveys flight informations and control decisions over wide-area networks.

1.1.3 Data Dissemination Challenges

The application scenario presented in the previous subsections 1.1.2 helps to comprehend that LCCIs are systems of systems that push far beyond the size of today's critical systems by every measure: number of dependancies, geographical extension, amount of exchanged and manipulated data or mass of interconnected computing elements. This introduces a revolutionary factor in the design of critical systems: dedicated networks cannot be used to convey information between the sensing and the control elements of the system. In fact, deploying dedicated networking wires and devices to interconnect the geographically-sparse systems composing an LCCI is too expensive and unfeasible, instead Internet connections are more suitable to be adopted as means to interconnect systems at wide area scale. This brings three main challenges on the data dissemination infrastructure, which is the object of this PhD dissertation:

- **Reliable and Timely Event Delivery** - Several experimental studies have demonstrated that WANs, such as Internet, exhibit a faulty behavior [8, 34], *i.e.*, several failures may occur and impose considerable message losses. Usually, it is possible to cope with such losses by using a well-defined reliability strategy, however, the gain of reliability is always achieved at the expense of delivery time, leading to a massive number of timing failures. Considering the timeliness requirement expressed at the beginning of subsection 1.1.1, LCCIs require that the adopted information dissemination infrastructure provides both reliable and timely message distribution, *i.e.*, messages must be delivered without any performance fluctuation even if the network may drop messages.

- **Scalable Event Dissemination** - The scale of future LCCIs represents a serious challenge when designing such infrastructures and must be taken into high consideration. The time to deliver a message does not have to be strongly affected by the scale of LCCIs, both in terms of devices composing the system (vertical scale) and the traffic generated by the data sources (horizontal scale).

- **Network Heterogeneity** - As previously stated, LCCIs are composed by heterogeneous routing domains and this variety implies that the network conditions are not the same all over the system. This means that one solution does not fit all: only one reliability strategy to guarantee event dissemination cannot be embodied into the middleware, but it has to be able to use the proper strategy depending on the experienced condition of the network, in terms of loss probability and burst length. Therefore, within the system, several reliability strategies can be used in the different parts of the systems that do not share the same network conditions.

LCCIs exhibit features and requirements that go beyond what current SCADA systems can provide. Specifically, information flow in SCADA systems follows the well-known *master-slave*, or client-server, interaction pattern, *e.g.*, when sensors feed a controller with monitoring data, they act as slaves while the controller is the master. Individual point-to-point and synchronous communications realized by master-slave interactions lead to rigid and static applications and make the development of dynamic large scale systems cumbersome. However, to carry out complex control functionalities in a cooperative matter among several geographically distributed control elements, LCCIs require more advanced and flexible interaction patterns rather than the master-slave one. In fact, to establish a master-slave interaction, the slave must know the identity and location of the master, and such dependency has to be defined when the system is deployed. However, due to the complexity of LCCIs, such knowledge may not be known at deployment time, but dependencies among the control elements of the infrastructure can also be dynamically established or disbanded when needed. In addition, previous research [35] has shown that it is hard for developers to keep track of many complex dependencies when deploying federated services. Rather than host-centric information dissemination(*i.e.*, the exact recipient is known) offered by SCADA systems, LCCIs require a more data-centric routing (*i.e.*, the content or type determines the destinations of the exchanged data), which has been proved to enforce plug & play connectivity among the components, to improve scalability of the information dissemination and to reduce configuration efforts. In the next section 1.2, we introduce a particular kind of data-centric middleware technology that is based on the publish/subscribe interaction model, and is considered the best candidate to be the reference middleware technology for

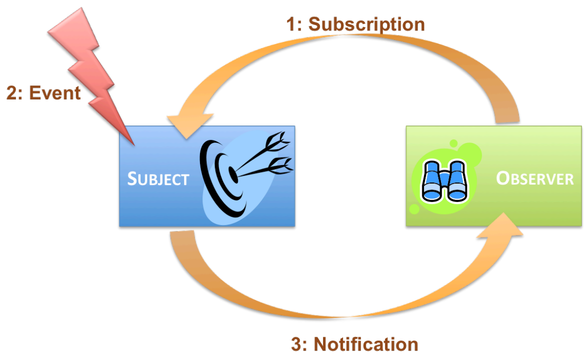


Figure 1.7: Operational schema of a generic event-based architecture

exchanging data in LCCIs[4].

1.2 Publish/Subscribe Systems

A middleware provides an abstraction of protocols and algorithms that allow an application to communicate between other entities in heterogenous distributed environments. Several middleware technologies have been developed during the years [36], and the most adopted interaction model is the *Remote Procedure Calls* (RPC) [37], where an application responds to received requests., and its derivative techniques such as the *Simple Object Access Protocol* (SOAP) [38]. Recently, a novel type of middleware is acquiring a progressive success: **Distributed Event-Based Systems** (DEBS) [39, 40]. In such systems, information is denoted as *event*, and the act of delivering is indicated as *notification*. Such architectures are based on the popular Object-Oriented design pattern called *Subject-Observer* pattern [41], illustrated in Figure 1.7: an object, called observer, is interested in knowing about events that occur in another object, called subject (the observer informs the subject on which

[4]OMG *Data Distribution Service* (DDS) [5] specification, which defines a standardized API for publish/subscribe services, has been selected by EuroControl as the technology to be used in the realization of the middleware that will interconnect all the European ACCs

events it is interested in using a subscription). Then, when a subject detects the occurrence of one of these events, it sends a notification to the observer. Such pattern has been further extended by the *Event Notification* pattern [42]: subject objects, called producers, and observer objects, called consumers, define, as part of their interface, the types of events that they are going to produce and to consume, and introduce decoupling between producers and consumers by using *Implicit Invocations* [43] of observer methods when events occur in the subject. This pattern, combined with the *Reactor* pattern, which dispatches incoming notifications to several handlers and the *Proactor* pattern [44], which uses asynchronous operations, made possible a novel interaction model that surpass the limitations of RPC and has been formalized in the *Publish/Subscribe* pattern [45]. Such pattern is widely adopted in most of the DEBSs so that it is the standard de facto for interactions within such middleware. Solutions adopting such interaction model are the focus of the rest of this section: subsection 1.2.1 provides general description of publish/subscribe services, which is further formalized in subsection 1.2.2; subsection 1.2.3 describes the intrinsic decoupling properties that characterize publish/subscribe services and enforce its scalability assurances, while subsection 1.2.4 presents the main notification architectures adopted in current solutions to glue together publishers and subscribers.

1.2.1 Generalities

A **publish/subscribe service** [12] is a middleware solution characterized by several processes that exchange messages and play two distinct roles within the systems: *Publishers*, which produce messages, and/or *Subscribers*, which consume the messages in whom they are interested. In fact, a process p_j receives only those messages that satisfy the k-th, namely $C_{j,k}$, of its active subscription predicates, contained in the set called Σ_j: if the message m_i matches the subscription $C_{j,k}$, then $C_{j,k}(m_i) \equiv \top$ and m_i is delivered to p_j, otherwise $C_{j,k}(m_i) \equiv \perp$ and m_i is not delivered to p_j. Given a message m_i, it is possible to define as its view, namely $viewof(m_i)$, all the processes that have to receive the message m_i:

$$viewof(m_i) = \begin{cases} V & \exists j,k : C_{j,k}(m_i) = \top \\ \perp & otherwise \end{cases} \tag{1.1}$$

where $V = \{p_j \mid \exists C_{j,k} \in \Sigma_j : C_{j,k}(m_i) = \top\}$. In addition, the basic architectural schema of a publish/subscribe comprises a third element in addition to publishers and subscribers: the so-called *Notification Service*. Such element plays the role of mediator between publishers and subscribers offering the following functionalities: 1) storing subscribers subscriptions; 2) receiving events from the publishers and buffering them; 3) dispatching received events to the interested subscribers by applying subscription-based filtering.

As said before, subscribers only receive events in which they are interested, and there are different ways of specifying such interest in a subset of the published events, which affect the architectures and algorithms adopted to implement the publish/subscribe platform:

1. *channel-based* [46]: publishers define a named channel to forward their notifications and subscribers only have to select a channel and connect to it in order to receive the events of interest;

2. *topic-based* [47]: publishers tag outgoing notification with a topic string, while subscribers uses string matching to detect the events of interest;

3. *content-based* [48]: subscribers express complex predicates on the content of events, and only those events that satisfy such predicates are delivered;

4. *type-based* [49]: subscribers express their interest only in events that present the same structure or event type.

A publish/subscribe can present one of such subscripting approach, such as CORBA Notification Service [50] that uses a channel-based method, or also a combination of them, such as OMG DDS [5] that adopts a combination of type-based and topic-based with the possibility to use content-based filtering at the subscriber side. In this dissertation, we focus only on topic-based publish/subscribe middleware.

1.2.2 Formal Specification

As illustrated in Figure 1.8, a process p_j performs a series of the following operations by interacting with the notification service, namely NS[5]:

1. *Publish*: a message m_i is published if p_j sends it to NS, and its view is not empty:

$$pub_j(m_i, p_j) \Rightarrow send_j(m_i, NS), \quad iff \quad viewof(m_i) \neq \perp; \tag{1.2}$$

2. *Notify*: p_j is notified by NS of a published message when a message is received by NS, and one of the subscriptions stored in NS and contained in Σ_j is verified:

$$not_j(m_i, p_j) \Rightarrow (recv_j(m_i, NS) \wedge C_{j,k}(m_i) = \top); \tag{1.3}$$

[5]For completeness we have drawn in Figure 1.8 also an additional operation at the publisher side. In fact, publishers may have the ability to *Advertise* future publications to the notification service, which adjusts itself to the expected flows of future events and informs subscribers of a new event type. Simple publish/subscribe systems do not have this feature, which is extensively use in content-based publish/subscribe to simplify the propagation of content-based predicates within the notification service [51], therefore, is not expressed in this specification.

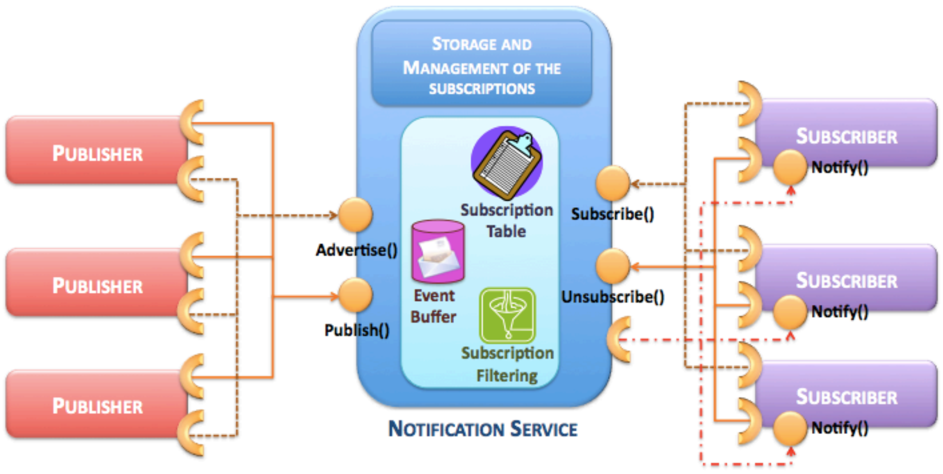


Figure 1.8: Schematic overview of a generic Publish/subscribe service

3. *Subscribe*: A new subscription $C_{j,k}$ is created by p_j and stored in NS:

$$sub_j(C_{j,k}, \Sigma_j) \Rightarrow (send_j(C_{j,k}, NS) \wedge \Sigma_j = \Sigma_j + C_{j,k}); \tag{1.4}$$

4. *Unsubscribe*: An existent subscription $C_{j,k}$ is erased from subscription table of NS:

$$unsub_j(C_{j,k}, \Sigma_j) \Rightarrow \Sigma_j = \Sigma_j - C_{j,k}. \tag{1.5}$$

A publish/subscribe service defined in terms of the previous operations has to satisfy the following two properties:

- *Safety*: a process p_j is notified of a message m_i by the notification service, because another process p_i has previously published it and p_j has expressed a subscription $C_{j,k}$ that is verified by m_i:

$$\exists p_j : t_n = not(m_i, p_j) \Rightarrow (\exists p_i : (t_p = pub(m_i, p_i) \wedge t_p < t_n) \wedge C_{j,k}(m_i) = \top); \tag{1.6}$$

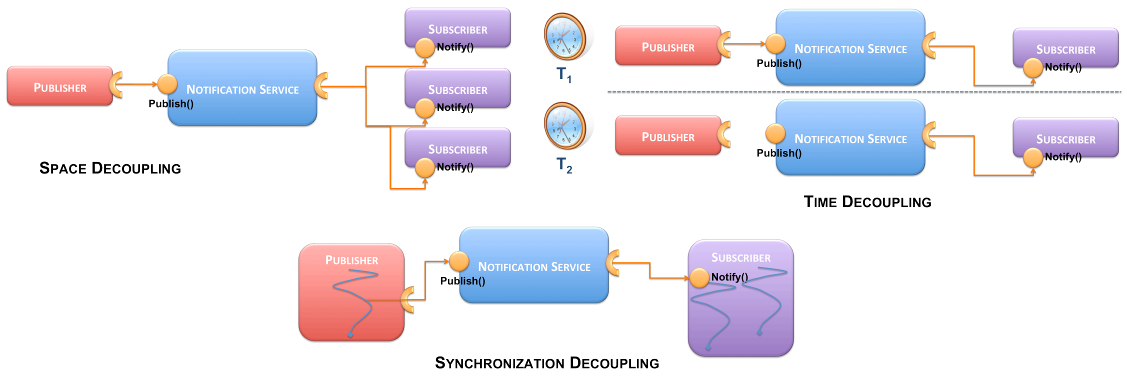


Figure 1.9: Space, time and synchronization decoupling provided by publish/subscribe middleware

- *Liveness*: if a process p_j expresses a subscription $C_{j,k}$, then every message that is published within the system and satisfies $C_{j,k}$, must be delivered to p_j:

$$\exists p_j, \exists C_{j,k} \in \Sigma_j \Rightarrow \exists m_i, \exists p_i : (pub(m_i, p_i) \wedge C_{j,k}(m_i) = \top) \Rightarrow not(m_i, p_j). \qquad (1.7)$$

The safety condition states that "something irremediably bad" will never happen within the systems [52], *i.e.*, a subscriber will never received an event in which it is not interested, and a notification service will never deliver an event that has never been produced by a publisher. On the other hand, the liveness condition demands that "something good" will eventually occur within the system [52], *i.e.*, a subscriber will always receive all the future notifications that satisfy its active subscriptions[6].

1.2.3 Decoupling Properties

As illustrated in Figure 1.9, publish/subscribe services offer three type of decoupling properties [12] due to the mediator role that is played by the notification service:

[6]Liveness condition formulated in Equation 1.7 does not impose any order on the notification deliveries since guaranteeing ordering properties demands additional algorithms that are not required by simple publish/subscribe services.

- *Space Decoupling*: the interactions among publishers and subscribers are completely anonymous: neither publishers neither subscribers know the participants to the communication. In fact, publishers do not directly send notifications to subscribers, but hold reference to the notification service and pass notifications to it, which is responsible to dispatch them to interested subscribers.
- *Time Decoupling*: publishers and subscribers do not have to be active at the same time. Specifically, publishers can produce events while subscribers are disconnected, while subscribers can get notified while the publishers are disconnected.
- *Synchronization Decoupling*: the execution of publishers is not blocked while publishing notifications, while subscribers are notified in an asynchronous manner (using callbacks) without interrupting some concurrent activities.

Decoupling production and consumption of notifications has proved to enforce scalability [53] since all explicit dependancies between the participants to event-based communications are removed. Due to the offered decoupling properties that make the resulting communication infrastructure well adapted to large-scale distributed environments, publish/subscribe solutions are considered the most suitable middleware infrastructure to glue together heterogeneous systems composing the LCCIs.

1.2.4 Notification Architectures

The notification service is usually accessed by the applications as a black box, conceptually centralized and viewed as local to each application, but it is implemented across several processes in order to split the load. The implementation of a notification service is based on

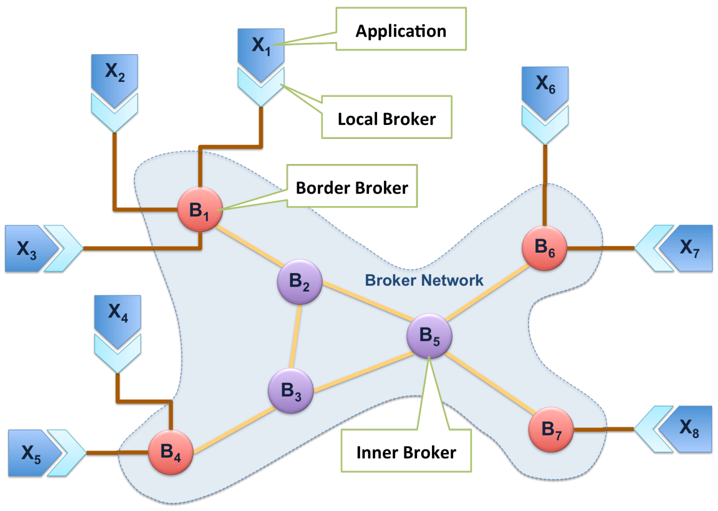


Figure 1.10: Broker network that implements the notification service

the concept of broker [39], and the general overview of the adopted architecture is provided in Figure 1.10. With respect to its position within the system and the performed duties, a broker can be classified in one of the following three types:

- *Local Brokers*, which provide means to access the notification service and are usually part of the communication library loaded into the applications;
- *Border Brokers*, which interconnect several local brokers and are placed at the boundary of the broker network.
- *Inner Brokers*, which represent the backbone of the notification service, forming an overlay network with border brokers and other inner brokers.

During the last decades different notification architectures have been proposed [54], and they can be classified in the taxonomy shown in Figure 1.10:

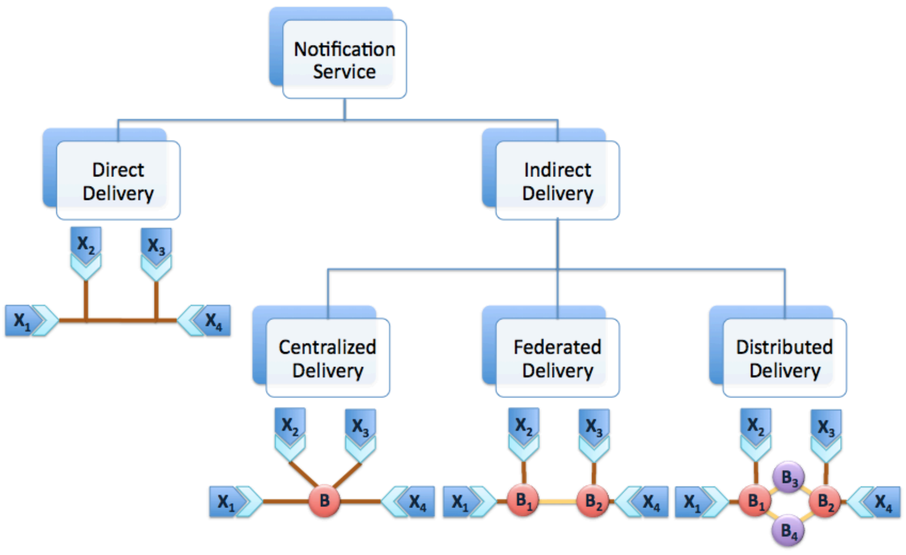


Figure 1.11: A taxonomy of notification service architectures

- *Direct Delivery*: each publishers takes the duty of delivery notification to each subscribers, without demanding a broker. The notification service is implemented using only local brokers that adopts IP Multicast [55] or peer-to-peer infrastructures [56] to deliver notifications, such as in all the implementations of the recent OMG standard called *Data Distribution Service* (DDS) [5].

- *Indirect Delivery*: the notification delivery is not directly handled by publishers, but such duty is entrusted to brokers. In fact, when the publishers directly take care of notification delivery, the overall publish/subscribe service lacks of scalability: if there are many notifications or subscribers to manage, the publisher performance are adversely affected since the overhead of sending notifications can completely overwhelm it. To treat such drawback, the notification duties can be shift from the publishers to a network of brokers, so that the notification service is composed of several brokers:

– *Centralized Delivery*: publishers and subscribers are tied together by a centralized broker, and the publish/subscribe service adopts a star topology. A practical example of such architecture is the communication of event suppliers and consumers through an event channel in the CORBA Event Service [46].

– *Federate Delivery*: centralized structures offer low availability guarantees since the overall system is vulnerable to broker failures, which can permanently disconnect publishers and subscribers. To improve the availability of the system, publishers ad subscribers can be grouped in different clusters, each one characterized by its own broker, which is directly interconnected with other similar brokers. The federation of event channel in the CORBA Event Service [57] is a well-known example of such architecture.

– *Distributed Delivery*: large scale systems demand strong scalability properties, and the most powerful architecture is to distribute notifications along a network of inner brokers that forms a tree or a mesh. Siena [58] and Padres [51] represent the most known publish/subscribe services implemented as a distributed network of brokers.

1.3 Reliable Publish/Subscribe Services

Since its inception, the publish/subscribe community has been more focused on scalable architectures, efficient delivery, and expressive subscriptions rather than reliable event dissemination. As a proof of this lack of attention on fault-tolerance issues, standardized and mature commercial publish/subscribe services do not address these issues at all, such as in the *Java Message Service* (JMS), or provide very basic mechanisms, such as DDS [5]. However, this status quo is changing as more and more publish/subscribe services have started to be used in application domains that expose stringent reliability requirements, *e.g.*, DDS implementation provided by Prismtech and named OpenSplice[7] has been chosen by RATP, *i.e.*, the Paris public transport operator, as the dissemination infrastructure in their system to control their railways and stations.

We have performed a survey of the available literature on reliable publish/subscribe services in order to measure the efforts spent on these issues by the international research community. We have collected all the papers published at international conferences and journals[8], counting how many of the proposed approaches focus on reliable publish/subscribe services in each year starting from 2000[9]. As shown by the black line in Figure 1.12, our study of the literature demonstrates that in the last decade we have witnessed a growing interest of the community in studying novel approaches to guarantee a reliable event dissemination upon networks and nodes that expose a faulty behavior. Despite such interest, there is not a universally-accepted definition of what a publish/subscribe service has to offer in order to be defined reliable, for this reason in subsection 1.3.1, we provide a formal

[7]www.opensplice.com

[8]We have taken under consideration the papers published by the following IEEE or ACM journals and conference proceedings over the last decade: IEEE Transactions on Software Engineering, IEEE/ACM Transactions on Networking, ACM Transactions on Computer Systems, IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Dependable and Secure Computing; the International IEEE Conference on Dependable Systems and Networks (DSN), the IEEE Symposium on Reliable Distributed Systems (SRDS), the IEEE Conference on Computer Communications (INFOCOM), the ACM/IFIP/USENIX International Middleware Conference (MIDDLEWARE), the ACM International Conference on Distributed Event-Based Systems (DEBS), the International Conference on Distributed Computing Systems (ICDCS).

[9]An approach may have been published in several papers in different years. *e.g.*, the Self-Stabilizing approach [14] has been published in 3 papers in 2005, 3 papers in 2006, 1 in the 2007 and 1 in 2008, we have counted it as a single paper in the median year of the distribution, *i.e.*, 2006.

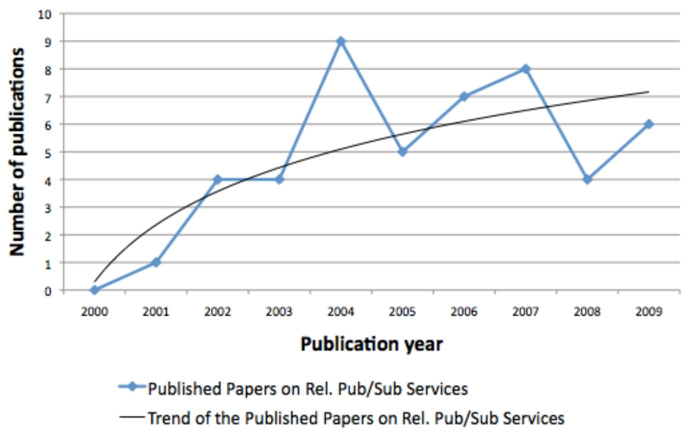


Figure 1.12: Efforts on reliability issues in publish/subscribe middleware

definition of the properties that make reliable a publish/subscribe service. In addition, we analyze in subsection 1.3.2 the plethora of failures that may affect a publish/subscribe service and we have used such failure model to study in subsection 1.3.3 if the start-of-the-art in reliable publish/subscribe services respects the definitions given in subsection 1.3.1, in order to find an approach can be adopted in the information dissemination in the context of LCCIs with respect to the requirements introduced in subsection 1.1.3.

1.3.1 Formal Specification

A publish/subscribe service is defined *reliable* if the message delivery is guaranteed despite that processes may fail and/or the network may be affected by several anomalies. Hence, taking advantage of the similarities of publish/subscribe services to group communication solutions[10] [59], we can formulate as reliable a publish/subscribe service that guarantees

[10]A Group Communication middleware is in charge of delivering messages within a group of processes. In case of channel-, topic- and type-based publish/subscribe services the membership to a certain group is defined by the submissions, e.g. given a topic all the subscribers interested to such topic and all the publishers that notify events belonging to this topic. On the contrary, content-based solutions do not present such static definition of groups, but the membership to a certain group of processes rather than to another one is dynamically defined by the content of the published events.

the following properties[11], considering a view, namely V, of the published message m_i:

- *Agreement*: if a non-faulty process p_j is notified, then all the other non-faulty processes are eventually notified:

$$\exists p_j \in V : not(m_i, p_j) \Rightarrow \forall p \in V : not(m_i, p); \tag{1.8}$$

- *Validity*: if a non-faulty process p_j publishes the message m_i, then at least one of all non-faulty processes is notified:

$$\exists p_j : pub(m_i, p_j) \Rightarrow \exists ! p_k \in V : not(m_i, p_k); \tag{1.9}$$

- *Integrity*: every non-faulty process p_j performs the notify of m_i at most once:

$$\exists p \in V : t_i = not(m_i, p) \Rightarrow \nexists t_j = not(m_i, p) : i \neq j. \tag{1.10}$$

The agreement property confers atomicity to an event dissemination operation since it apply a "all or nothing" prospective to the notification delivery: a message is delivered to all the interested subscribers or to none of them. On the other hand, the validity property guarantees that if a publisher produce a notification and its related view is not empty, then it will reach at least one interested subscriber despite of possible failures (removing the "nothing" prospective allowed by the previous property). While, the integrity property assures that no duplicate notifications of a given event are delivered to the application. In addition to this properties, since critical systems, such as LCCIs, exhibit real-time constraints jointly to fault-tolerance, *i.e.*, a notification delivered out of temporal deadlines is considered as

[11]It is possible to have a *Uniform* specification of these three properties if they state the expected behavior even of fault processes. However, in this thesis, we do not consider the uniform form of such properties

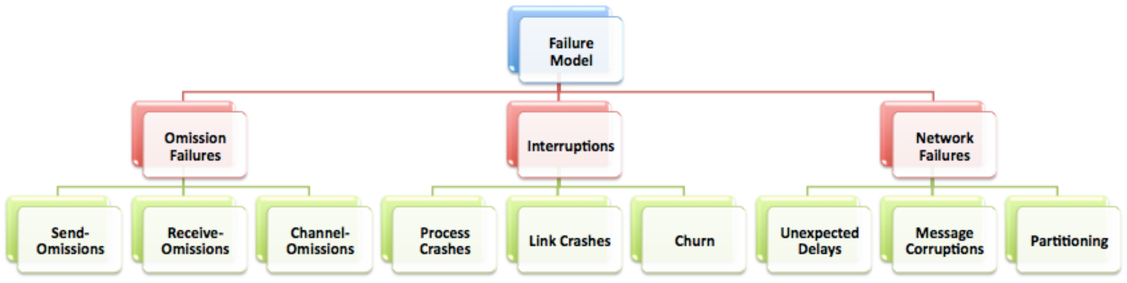


Figure 1.13: Model of the failures that may affect the correct delivery of notifications

lost. Therefore, a publish/subscribe service to be used in such systems has to verifies an additional property:

- *Timeliness*: given a deadline Δ, all non-faulty processes are notified of a published message before Δ is expired:

$$\exists \Delta, \exists p_j : pub(m_i, p_j) \Rightarrow \nexists p_k \in V : not(m_i, p_k) > \Delta. \tag{1.11}$$

I.e., the delivery time of every published event exhibits a known upper bound Δ.

1.3.2 Failure Model

A publish/subscribe system is composed of several nodes that run processes[12], which play the role of publishers/subscribers or brokers, interconnected by communication channels. Both processes and channels may expose faulty behaviors, *i.e.*, they behave away from what is considered correct or desirable behaviors, and they need to be carefully treated in order to achieve reliability by satisfying the previous properties. As shown in Figure 1.13, there are three main classes within which failures that may affect processes and/or channels can be grouped in the following classes:

[12]Without loss of generality, we assume that a node only runs a single process. So, in this dissertation the terms "node" and "process" are used interchangeably.

- *Omission Failures*: an element, *i.e.*, process or channel, fail to perform operations that they are supposed to do. When disseminating a message, the following type of omissions can occur are the following ones:
  - *Send-Omissions*: a process fails to perform a send operation, *i.e.*, a message is lost before it leaves the network interface. Such omissions are caused by a lack of space in the sending buffers at the operating system and/or network interface.
  - *Receive-Omissions*: a process fails to obtain a message even if it has been successfully received from the network. Since this is dual to send-omissions, also receive-omissions are caused by a lack of space in the receiving buffers.
  - *Channel-Omissions*: a channel drops messages exchanged between a pair of processes due to hardware faults, congestion and/or lack of space in the buffers of traversed routers.
- *Interruptions*: processes and links suddenly stop working:
  - *Node crashes*: nodes stop to produce notifications or react to incoming messages due to countless types of hardware/software failures, *e.g.*, aging phenomena, software bugs and/or hardware malfunctioning.
  - *Link crashes*: links experience complete loss of connectivity, *i.e.*, packets are always lost for a certain time, which is not necessarily permanent but may dynamically appear and disappear.
  - *Churn*: nodes suddenly join and leave the system.

- *Network Failures*: incorrect behavior exhibited by the network:
  - *Unexpected Delays*: the time needed to exchange a message along a channel is greater than the maximum expected delivery time. Congestion or overloading of some router are the causes of such behavior.
  - *Message Corruption*: the content of a message can be corrupted while been exchanged over a channel. Message corruption has a variety of causes, but mostly it depends on environmental conditions that can interfere with data transmission, especially when dealing with wireless transmission methods.
  - *Partitioning*: the network may be segmented in several sub-networks, within which messages can be exchanged but among which messages are lost, due to malfunctioning network devices or broken network connections. Usually, the network stays partitioned for a limited period, then the cause of the network partitioning is repaired and the connectivity within the network is restored by merging the several sub-networks originated by the partitioning.

1.3.3 Analysis of the state-of-the-art

With respect to the taxonomy presented in the previous subsection 1.3.2, we have analyzed which faults are handled by the reliable publish/subscribe services surveyed at the beginning of this section. The results of this analysis are shown in Figure 1.14 and in Table 1.1. In our analysis, the overall class of omissions are simply indicated as losses since dropped messages are detected at the subscriber side without any discrimination of their origin, *i.e.*, they are sender-, receiver- or channel-omissions. Moreover, we have not paid a lot of attention to

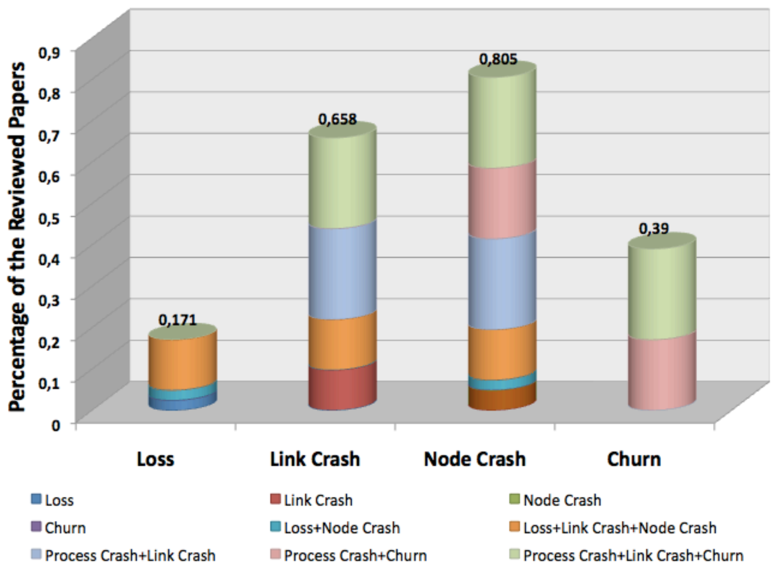


Figure 1.14: Failures treated by reliable publish/subscribe services proposed in the reviewed papers

network failures since less than 3% of the reviewed approaches deal with them. The reason may be that unexpected delays and message corruptions are considered at the application level as message omissions (by applying in the first case proper timeouts, while in the second case Cyclic Redundancy Check (CRC) codes [60] can be used to detect a corruption and to discard corrupted messages), while partitioning is treated as a special case of link crashes (in fact, all the approaches handling network partitions, treat also link crashes). Therefore, the four failures that result more considered in the literature are message losses, link and node crashes and churn.

Traditionally, reliable notification has two related but distinct meanings [61]: (*i*) it can refer to the resiliency of the message dissemination infrastructure to the volatile nature of its components due to crashes or churn, and (*ii*) it can also refer to the reliable delivery of multicast content from providers to consumers despite of possible losses. We have noticed

Table 1.1: Analysis of the reliable publish/subscribe services available in literature

Reviewed Approaches:		41		
Network Failures	Losses	Link Crashes	Node Crashes	Churn
3	7	27	33	17
Dealing with volatile elements:		31		

Total Approaches	Losses	Link Crashes	Node Crashes	Churn
1	√			
4		√		
2			√	
1	√		√	
5	√	√	√	
9		√	√	
7			√	√
9		√	√	√

unbalanced interested of the community on these two aspects. As evident in Figure 1.14, many efforts have been spent to address faults that can lead to inconsistent topologies within the broker overlay (*i.e.*, the first of the previously-introduced two aspects). Specifically, about 76% of the analyzed approaches spend efforts to recover the connectivity within the publish/subscribe service dealing with the volatile nature of link and processes to reconstruct the lost connectivity among publishers and subscribers. If we carefully look Table 1.1, we notice that 80% of the reviewed approaches treat node crashes, making it the most studied fault. Specifically 34% of these systems focus on the case in which the crashed node hosted a process that acted as a border or inner broker, which is critical in the system since it glues together publishers and subscribers and its failure can cause loss of information and connectivity. While, only a smaller number of the analyzed systems, *i.e.*, 39%, handle also churn. This may be due to two reasons: 1) often these systems are built on peer-to-peer infrastructures, which gracefully manage these dynamics[13]; 2) this fault is treated for free, without a particular solution. In fact, a close scrutiny of Figure 1.14 reveals that systems dealing with churn also handle node crashes. In fact, churn can be seen as a special case of node crash, so all the techniques adopted to cope with node crashes can also be used in this other case. On the other hand, the second most studied class of faults is the link crash, specifically in about 66% of the systems, but only 15% do not cope also with node crashes, while the rest of them treat both link and node crashes. Something similar also happen considering node crashes: only 30% of the systems that handle node crashes, do not deal with link crashes. In fact, 56% of the reviewed systems jointly handle link and node crashes since they use routing techniques, such as self-stabilizing reconfigurations [14] or redundant paths within the broker network [17], that are able to circumvent the crashed element without leading to persistent disconnections within the system. At this point, the question that we have made to ourselves is: do these systems provide the reliability properties of agreement, validity and integrity if only link and node crashes occur within the system? Unfortunately, the answer is not affirmative: only 25% of such systems presents techniques to tolerate losses caused by the temporary disconnection due to crashes. In fact, the focus of the rest of these systems is much more on recovering the connectivity among the different elements composing the publish/subscribe system rather then recovering the dropped messages. Even if recovering lost connectivity is important, also tolerating messages dropped due to the temporary loss of connectivity do not have

[13]In fact, the ones that do not care about churn adopt a static broker overlay rather than a peer-to-peer infrastructure.

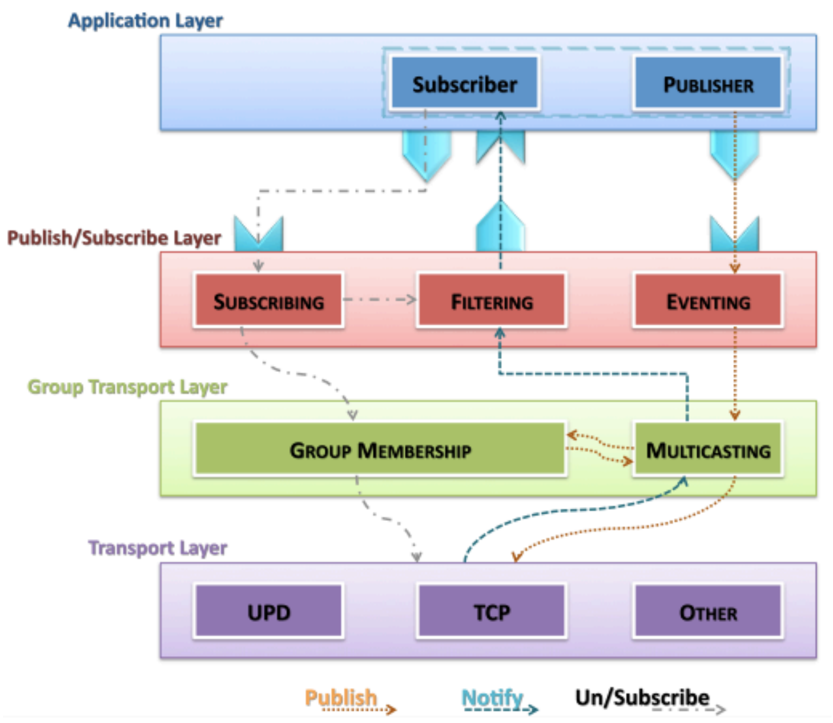


Figure 1.15: Layered Organization of a publish/subscribe service

ignored, otherwise agreement can not be obtained.

The reason behind why only 17% of all the proposed reliable publish/subscribe services handle message losses may be found on the consideration that publish/subscribe services are designed on top of an *multicast protocol*. As shown in Figure 1.15, below a publish/subscribe layer that offers to the applications the generic functionalities of publish(), un/subscribe() and notify(), there is layer that provide a protocol to manage the membership of the application to certain groups, and a multicast protocol to disseminate and receive messages exchanged within the interested groups. Studying how to cope with network anomalies has hitherto not been the focus of the community since it was deemed to be easily addressed using a reliable multicast protocol, such as one of the reliable multicast protocols developed during the last twenty years [62, 63, 64]. However, loss-tolerance is not trivial and represents a crucial aspect to guarantee the properties expressed in subsection 1.3.1. In the next chapter, we have analyzed the literature related to loss tolerance in multicast communications in order to point out the strengths and weaknesses of all the available approaches, and understand if the ones adopted in the few publish/subscribe services that tolerate losses and crashes are suitable for the satisfaction of the introduced reliability properties.

Chapter 2

Loss Recovery for Reliable Group Communication

This chapter presents approaches to provide a reliable communication among the members of a group and evaluates their strength and weakness. Specifically, the focus is on how tolerating message losses caused by network pathologies and/or crashes of link and nodes forming the dissemination infrastructure. The chapter is structures in three parts: the first one introduces the two different architectural perspective to design a group communication protocol, the second part discusses the state-of-the-art on loss-tolerant multicasting, while the last part analyzes the overviewed techniques to assess which one is the optimal for achieving a reliable and timely event dissemination.

2.1 Architectures to design a group communication system

Multicast communication has been an active research topic in the last decades [65, 66, 67], and in the years several protocols have been developed to support a reliable dissemination of messages among the members of a group [62, 64]. Such protocols are traditionally grouped in two main classes [68]: **Transport-Level Multicast** (TLM) and **Application-Level Multicast** (ALM). As the name suggests, TLM protocols devise the mechanism for multi-point content delivery at the Transport Layer of the ISO/OSI stack by adopting IP Multicast [55, 69]:

1. each group has assigned a multicast address, which falls in the range between 224.0.0.0 and 239.255.255.255 according to RFC 3171, and a sender uses it to forward a message

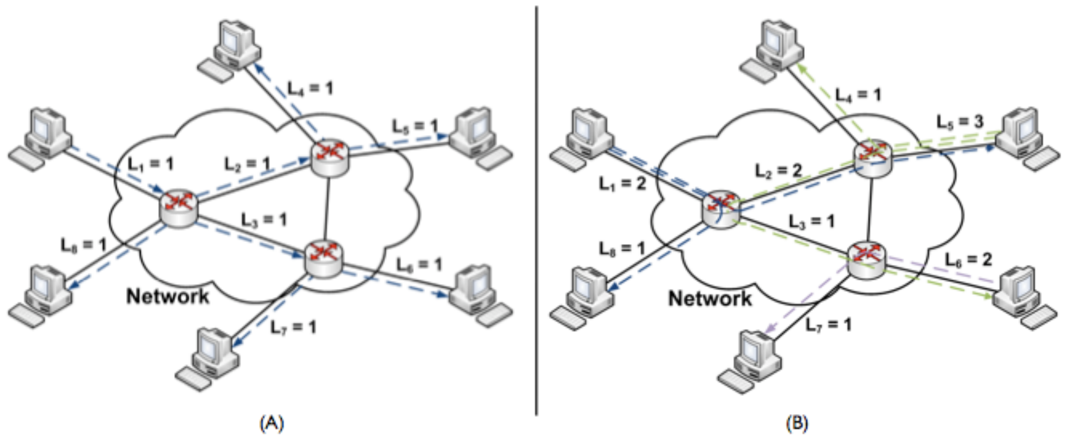


Figure 2.1: Analysis of the efficiency of the two different perspectives to device Reliable Multicast: a) Transport-Level Multicast and b) Application-Level Multicast

to all the members of the given group;

2. routers take the duty of replicating an incoming message in order to deliver it to interested destinations through several outgoing channels.

On the other hand, ALMs do not use IP Multicast but implement the multicasting service at the application layer [67]. As shown in figure, end-systems belonging to a group are interconnected using an overlay network and the duty of replicating a message when it has to be dispatched over several distinct outgoing links is carried out by end-systems rather than routers. The key architectural question is which paradigm is more suitable to implement an effective multicasting: "to be or not to be" at the transport layer? Unfortunately, there is no simple answer since the choice strongly depends on the requirements that the multicast protocol has to address. In fact, an architecture perspective may fit some use cases but does not other ones.

One measure to compare different solutions is investigating their *efficiency*, namely η, which in the case of multicast services can be evaluated in terms of *link stress*, *i.e.*, the number of messages exchanged through a link during a multicast session:

$$\eta = \frac{N}{\Sigma_i L_i} \tag{2.1}$$

where N is the number of links traversed during a given multicast session and L_i is the link stress on the i-th link. If we consider the multicast communication performed using IP Multicast shown in Figure 2.1A, the link stress for each link is always equal to 1, so the efficiency of the overall communication is 1. On the other hand, an ALM is less efficient than the one using IP Multicast. In fact, if we consider Figure 2.1B, we can notice that some links experience a link stress greater than 1, and the efficiency of the overall system is equal to 0,692 (with a reduction of almost 40% even if the system topology was very simple). Therefore, IP Multicast provides a wiser use of the network resources, by reducing the traffic, and better performances that any ALM protocol. Although the efficiency analysis concluded that IP Multicast outperforms ALM, it is a mistake to think that it represents the best solution for every application scenario. In fact, the adoption of IP Multicast has been dogged by several drawbacks. On one hand, IP Multicast exhibits severe deployment limitations over Internet [70]. In fact, although all the commercial routers that are currently marketed support IP Multicast, it can be used only in network domains that are managed by a single organization or in local area networks. Internet Service Providers (ISPs) are reluctant to enable data dissemination through IP Multicast in their domains in order to reduce router load and protect against unwanted traffic [66]. In addition, IP Multicast requires substantial changes to the networking infrastructure, *e.g.*, replacement of older hardware with a new one that offer a native support to multicast, and their costs are not easily supported by the business model of most all the ISPs. Therefore, currently Internet is characterized by few and scattered "islands" where IP Multicast is supported, while most of all the other portions do not provide it. In addition, IP Multicast forces routers to maintain per-group state [68], *e.g.*, members to the group, that is highly variable over time. This state management clearly violates the "stateless" perspective of the IP protocol, and also introduces strong complexities and scaling issues in a wide-area scenario such as Internet. Despite their intrinsic inefficiency, ALMs present neither the deployment issues neither the scaling limitations that affect IP Multicast over wide area networks. The effective deployability and scalability of ALMs on today's Internet have been proved by the incredibly success of file sharing applications such as Napster and Kazaa, which adopts ALM solutions to allow downloading a single file by multiple, geographically distributed, users per time [67]. So, ALMs are largely adopted to address the matter of multicasting messages among several destinations over wide-area networks, while, TLMs have been successfully used in clusters and datacenters [71], where IP Multicast is likely to be supported.

2.2 Approaches to provide loss-tolerance guarantees

The common practice when designing a reliable multicast is to architect it in two distinct layers: a lower level consists of a protocol to deliver messages within a given group, while, an upper level is built on top of it with the duty to implement suitable means to support specific quality-of-service properties. As previously stated, TLM solutions are developed on top of IP Multicast that offers only best-effort guarantees for the message delivery, following the architectural choice to provide reliability assurances at transport level rather than at network one in the ISO/OSI stack. On the other hand, mostly all the ALM approaches use UDP connections to exchange messages among end-systems, while few of them, such as RMX [72], use TCP connections. TCP protocol is not largely adopted in ALM since, as proved by several studies [73], multicast over overlay networks of TCP connections is affected by scalability concerns, in terms of throughput, buffer requirements and delivery latency. In addition, TCP protocol support per-link reliability assurances, *i.e.*, a message exchanged along a TCP connection between a pair of end-systems is assured to be delivered. However, the reliability properties previously introduced in subsection 1.3.1 are per-group valid, *i.e.*, all the members of a group needs to receive a given message despite of possible failures. Even if delivery is guaranteed along a connection, using only TCP protocol does not guarantee that messages are correctly delivered to all the members, *e.g.*, an end-system may crash, leading to a disconnection, and messages are not delivered to a portion of the group. The rest of this section discusses of the techniques available in literature that TLM and ALM solutions adopt to tolerate message losses.

2.2.1 Loss Recovery In Transport-Level Multicast

The most popular, and easy to implement, solutions to tolerate message losses are **Automatic ReQuest Repeat** (ARQ) schemes that adopt temporal redundancy, *i.e.*, using retransmissions to recover dropped messages [62]. These approaches exhibit a reactive nature since recovery actions are taken only after a loss has been detected. This means that if

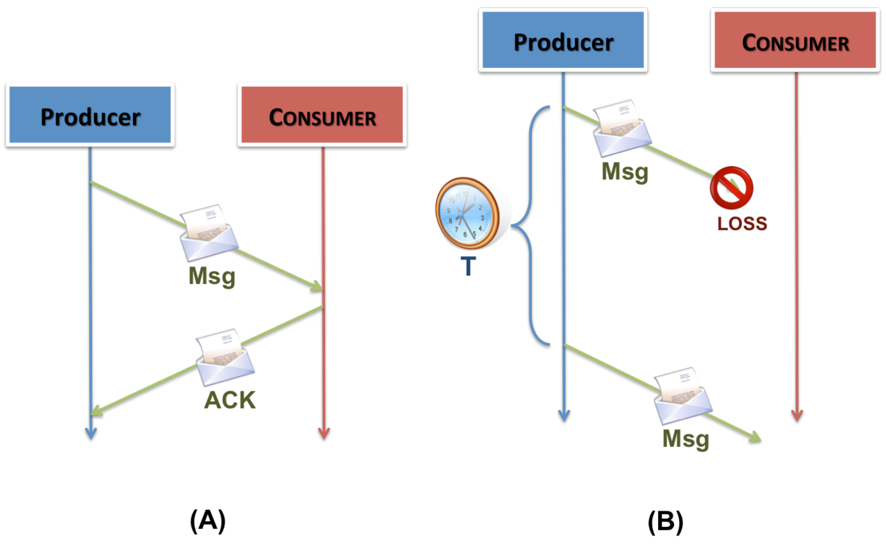


Figure 2.2: Simple ARQ scheme

we define the probability to loss a message msg_i as π_i, then the time to deliver the message to a given destination, namely λ_i is equal to:

$$\lambda_{ARQ,i} = (1 - \pi_i)\delta_i + \pi_i(D_i + R_i) \tag{2.2}$$

where δ_i is the latency needed by messages to reach the destination if there are no failures on the channel (namely failure-free delay), D_i is the time needed to detect that a message has been lost and R_i is the time required to recover a dropped message after its loss has been detected. Reactive techniques are characterized by a time to deliver a message when a failure occurs, *i.e.*, $D_i + R_i$, that strongly depends on the network behavior and is not predictable. Without loss of generality, to prove this statement let consider a simple ARQ scheme shown in Figure 2.2: a producer sends a message to a consumer, which reply with an ACK message to notify the producer of the correct delivery of the message, as shown in the left side of Figure 2.2; if the ACK message is not received before a timeout expires, the producer retransmits the message, as illustrated in the right side of Figure 2.2, which have a certain probability to be correctly delivered. For this simple scheme, the dissemination latency of a given message msg_i can be formulated considering Equation 2.2:

$$\lambda_{ARQ,i} = (1 - \pi_i)\delta_i + \pi_i(\psi_i \cdot T + \delta_i) = \delta_i \cdot T \cdot \psi_i \cdot \pi_i \tag{2.3}$$

where ψ_i is the number of consecutive messages lost before m_i could successfully reach the consumer, while T is the value of the adopted timeout, which has to be twice greater than δ_i, *i.e.*, $T \geq 2 \cdot \delta_i$ in order to avoid unneeded retransmissions. The mean time to deliver a message is the following:

$$\Lambda_{ARQ} = E[\lambda_{ARQ,i}] = \Delta \cdot T \cdot \Psi \cdot \Pi \approx 3 \cdot \Delta \cdot \Psi \cdot \Pi \tag{2.4}$$

where Δ is the mean time to deliver a message in a failure-free scenario, Ψ is the average burst length, *i.e.*, the mean number of consecutive messages lost by the network, and Π is the probability that a message is lost by the network. While Δ is slightly stable in a failure-free scenario, and depends on the path that interconnect producer and consumer, Ψ and Π are strongly affected by the network conditions and are not known a priori. In fact, prior studies [74, 75] have proved that loss statistics are not constant during the day due to the higher amount of traffic, especially HTTP requests, corresponding to intense web surfing activity during specific hours of the day.

Using spatial redundancy rather than the temporal one, *i.e.*, reliability means are taken proactively even if losses do not occur, allows achieving a more timely multicast even in case of failures. The most adopted method to implement spatial redundancy in TLM is based on **Forward Error Correction** (FEC) [18]: the multicaster forwards additional data so that the destination can reconstruct the original information even if losses occurred. In case of FEC-based recovery, the time to deliver a message can be formalized as follows:

$$\lambda_{FEC,i} = (1 - \pi_i)(\Omega_{send,i} + \delta_i) + \pi_i(\Omega_{send,i} + \delta_i + \Omega_{recv,i}) = \Omega_{send,i} + \delta_i + \pi_i \cdot \Omega_{recv,i} \quad (2.5)$$

where $\Omega_{send,i}$ is an overhead needed to build the additional data from the message m_i, while $\Omega_{recv,i}$ is the overhead required to construct the received data in order to obtain m_i when some messages have been dropped. Therefore, the mean time to deliver a message can be formulated as follows:

$$\Lambda_{FEC} = E[\lambda_{FEC,i}] = \Omega_{send} + \Delta + \Pi \cdot \Omega_{recv} \quad (2.6)$$

It is important to notice that the overhead imposed by FEC at sender and receiver side does not depend on the network conditions (*i.e.*, Ω_{send} and Ω_{recv} only depends on the adopted redundancy degree), making the multicast time more predictable. Although FEC assures timely delivery even in case of dropped messages, it do not exhibit strong reliability guarantees. In fact, it the number of the lost messages exceeds the redundancy degree applied by the multicaster, the destination is not able to reconstruct the original message, which is irreparably lost. For this reason, in literature there are several approaches that somehow combine the previous two strategies in order to jointly achieve strong reliability and timeliness properties. In this section, the main techniques belonging to these three classes are introduced and their points of strength and weakness are discussed.

2.2.1.1 Centralized Retransmissions

The first approaches based on temporal redundancy follow a centralized perspective, *i.e.*, the duty to store and retransmit messages is centralized in a single element of the system.

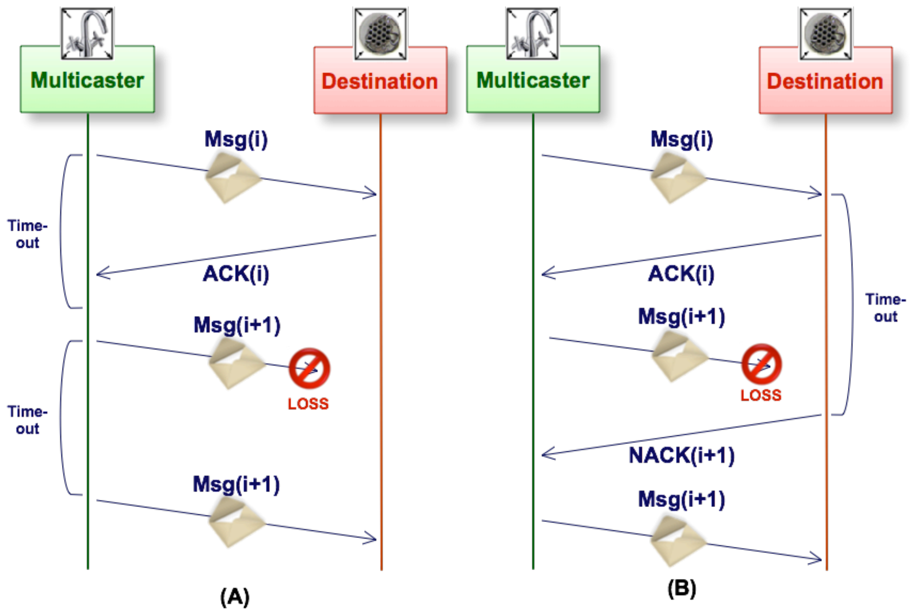


Figure 2.3: Execution of a sender-initiated protocol with a) sender-driven retransmissions and b) receiver-driven retransmissions with receiver-based timeouts

The prior protocols of this type adopted a naive implementation, called **sender-initiated retransmissions** [76], which is illustrated in Figure 2.3, by mimicking the behavior of the TCP protocol: the multicaster holds the responsibility to guarantee reliable message dissemination to the members of a given group. Specifically, messages are stored at the sender side and forwarded to all the interested destinations. Then, each receiver has to reply with a positive acknowledge (ACK) in case of the successful reception. Last, the source can release the memory associated to a give message only when all the destinations sent ACKs related to it, and dropped messages are detected by the source through timeouts (*source-driven retransmissions*) on the expected time to collect all the ACKs, and retransmissions are triggered. Such approach, however, is not scalable for two main reasons: on one hand, sender may be overloaded or channels may suffer of congestion and packet collision due to a overwhelming number of ACKs that reach the sender[1], on the other hand, a centralized timeout is not optimal when several heterogeneous destinations, *i.e.*, each one characterized by a different time to send back ACKs, are involved in a multicast session. As suggested in [77], a better approach is to allow *receiver-driven retransmissions* by combining ACKs with negative acknowledges (NACKs): a destination keeps sending ACKs when a message is successfully delivered, while it sends NACKs to trigger a retransmission. There are two possible approaches to detect a message loss: *Receiver-based Timeout*, *i.e.*, a NACK is sent when the destination did not receive a given message before the expiration of its local timeout, and *Sender-based Sequencing*, *i.e.*, sender gives a unique sequential identifier, namely senquence_id, to produced messages, so that a destination forwards a NACK when the sequence_id of the received message is not equal to the expected one. The latter approach is more simple to implement, but it presents higher recovery time. Last, in order to reduce the ACK implosion, ACKs can be grouped so that they refer to a group of messages rather than a specific one.

Sender-initiated protocols have the drawback that a sender needs to collect ACKs from all the destinations in order to free memory. This requires the complete knowledge of the receiver set, and in most of cases such knowledge is difficult, or impossible, to obtain. To address such issue, a different kind of solutions, called **receiver-initiated retransmissions** [78] and illustrated in Figure 2.4, has been proposed. Such protocols are characterized by the absence of ACKs and destinations can reply only with NACKs to notify cases of dropped messages. The main weakness of receiver-initiated recovery strategy is that it is

[1]This phenomenon is called "*ACK implosion*".

troublesome to define when the sender can free the memory used to store a given message since it is hard to know when all the destinations have received it. Mostly all approaches adopt an optimistic perspective by setting a timeout for the storage of a message and freeing related memory at its expiration. However, the tuning of this timeout is critical since a dropped message can be never recovered if a NACK is received later then the expiration of the timeout. On the other hand, a message can be retransmitted several times since a sender can receive multiple NACKs at different time instances[2]. To treat such matter, NACKs can be avoided by such scheme: a destination waits the expiration of a timer[3] before sending a NACK, then it forwards the NACK not only to the senders, but also to all the other destinations. If a destination has been reached by a NACK for a message that it has not yet received and for which it has started a timer to send a NACK, then it stops the timer and behaves as it has sent a NACK. So, even if a message did not reach several destinations, the sender may hopefully receive only one NACK. Receiver-initiated protocols with such *NACK avoidance scheme* are sometimes referred as *RINA* protocols [78], and they have been demonstrated to provide better performance than the basic receive-initiated protocols [62]. Even if RINA protocols exhibit better scalability than the previous sender-initiated ones, they can be successfully used for a limited audience. In fact, NACKS and retransmissions still needs to be forwarded to the entire group. If only few destinations experience losses due to heavy lossy links, an abundance of NACKS and repair messages cannot be completely avoided, with the consequent resource consumption for other loss-free

[2]This phenomenon is called "*NACK implosion*".

[3]The value assigned to the timer may be or a function of the receiver's distance from the sender, or a random value, or a combination of both methods, such as in Scalable Reliable Multicast [77]. Regardless of the chosen method, two destinations do not have the same timer value.

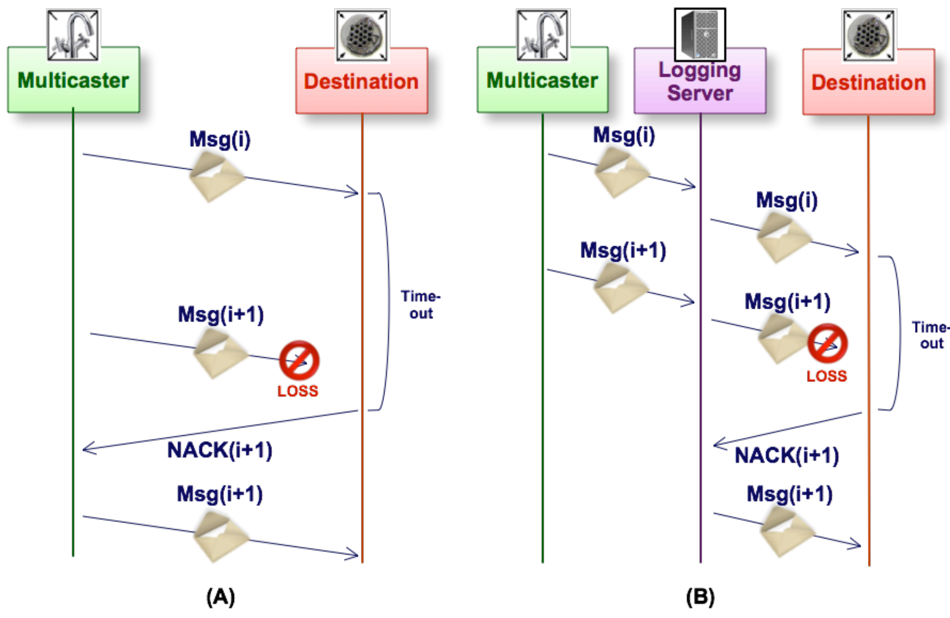


Figure 2.4: Execution of a receiver-initiated protocol a) without and b) with a logging server

destinations[4].

A final consideration that we can make regarding such centralized solutions is that they result weak against crashes of the multicaster. In fact, if the multicaster is unavailable, destinations cannot recover dropped messages. To enforce the reliability assurances even in case of multicaster crashes, all generated messages are stored in logging servers [79], which run on stable machines whose probability to crash is negligible, rather than at the multicaster. Logging servers works as brokers interleaved among the source and the destinations, as illustrated in Figure 2.4b: the source forwards messages to the logging server though TCP connections, then the logging server stores the received messages and deliver them to proper destinations though a IP Multicast session. Destinations contact the logging server to recover dropped messages using a receiver-initiated protocol.

[4]This has been referred as the ”crying baby problem” [79]

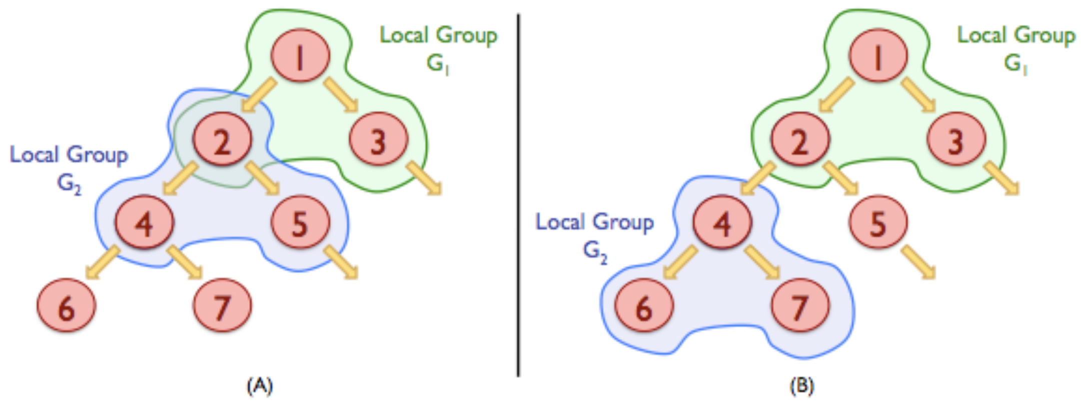


Figure 2.5: Tree-based organization of the members within a group: a) overlapping groups and b) separated groups

2.2.1.2 Hierarchical Retransmissions

Hierarchical protocols [80] aim to improve the scalability feature of reliable TLM by introducing an organization of the destinations[5] in groups and distributing the retransmission duties over the groups. Specifically, all the destinations are structured in a so-called *ACK Tree*, which prevent them to directly contact the source for retransmissions. Each local group within the ACK Tree is characterized by a leader, which holds the responsibility to store received messages and perform retransmissions, while the other members, namely followers, request retransmissions of the messages that they do not have received. As it can be noticed in Figure 2.5, local groups may exhibit overlaps, *i.e.*, a destination can belong to two distinct groups, and destinations can have different rules, *i.e.*, node 2 is the leader of local group G_2 while it is a follower in local group G_1, or local groups do not overlap and the leader of local group is in contact with one of the followers of one local group at higher

[5]Also the scalability of a receiver-initiated protocol that uses a logging server can be improved by placing multiple servers within the system and organizing them in a hierarchy in order to split the retransmission responsibility among them.

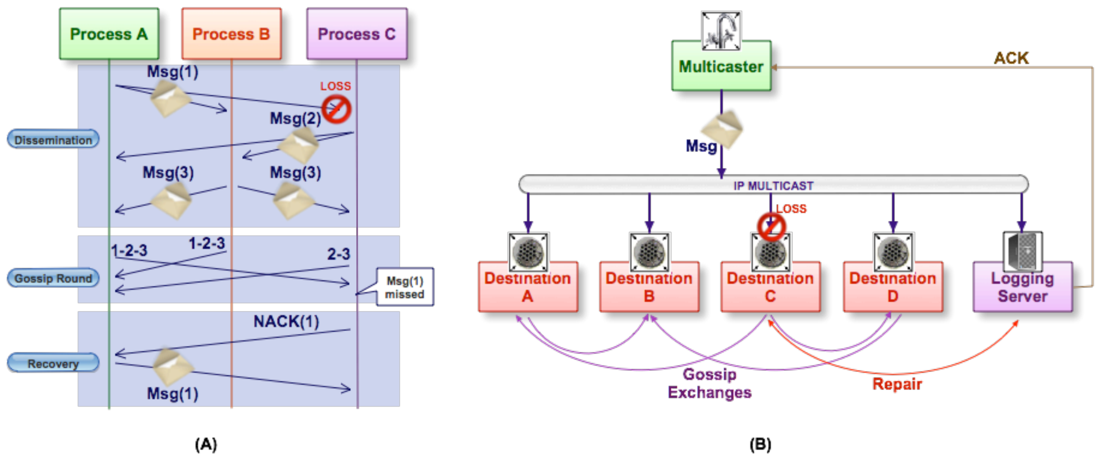


Figure 2.6: Execution of gossiping a) in a pure fashion and b) with a logging server

level in the hierarchy. Locally at each group a sender-initiated or receiver-initiated scheme can be applied. Although introducing a tree-based organization among the members of a group enforces scalability, such mechanism sensible increase latency, especially at the lowest level of the tree, to successfully receive a message, leading to larger timeouts at the sender and delaying loss discovery [81].

2.2.1.3 Distributed Retransmissions

A criticism to all the previous ARQ schemes is that strong reliability properties are obtained at the expenses of tolerating unstable and unpredictable performance and scalability limits. A solution to reduce these drawbacks is distributing the retransmission duty among all the destinations, and this can be realized by using the so-called **Gossiping** [82], illustrated in in Figure 2.6A. Specifically, all the destinations receive messages though IP Multicast sessions, and periodically commerce a gossip round. During this round, a given destination sends to other randomly-chosen destinations a summary of the received messages. When a destination detects an inconsistency by comparing the received summary and its own history of received messages, it sends a NACK to the sender of the summary to solicit a retransmission and converge to the same history of received messages. Processes would gossip about a message for a fixed number of rounds, then it is no more included in the exchanged summary and it is dequeued from the process buffer (*i.e.*, executing a garbage collection of the message buffer). In fact, gossiping prioritized the recovery of recent messages, and when a message is too old, it gives up to recover it and notifies the application that the message has been lost. The choice of how many round a message will be gossiped is crucial since after a message has been deleted by the garbage collector, the gossiping fails to recover the message. These circumstances are avoided by the protocol proposed in [83] by combining the logger-based approach of [79] and the gossiping of [82], as shown in Figure 2.6B. In fact, it has a logging server that stores the messaged produced by the multicaster, when the gossiping fails, then the destination can contact the logging server to recover the dropped message. Gossiping allows realizing a scalable reliable multicast by weakening the reliability assurances. In fact, the randomness nature of the protocol guarantees the agreement property only in a probabilistic manner, *i.e.*, there exists a certain, high, probability that all the destinations reach the same view of the exchanged messages.

2.2.1.4 Sender-initiated FEC

As previously stated, Forward Error Correction involves multicasting redundant packets along with data packets, so that the destination can recover lost packets without requiring any retransmission to the sender or to any other destination. The execution schema of a

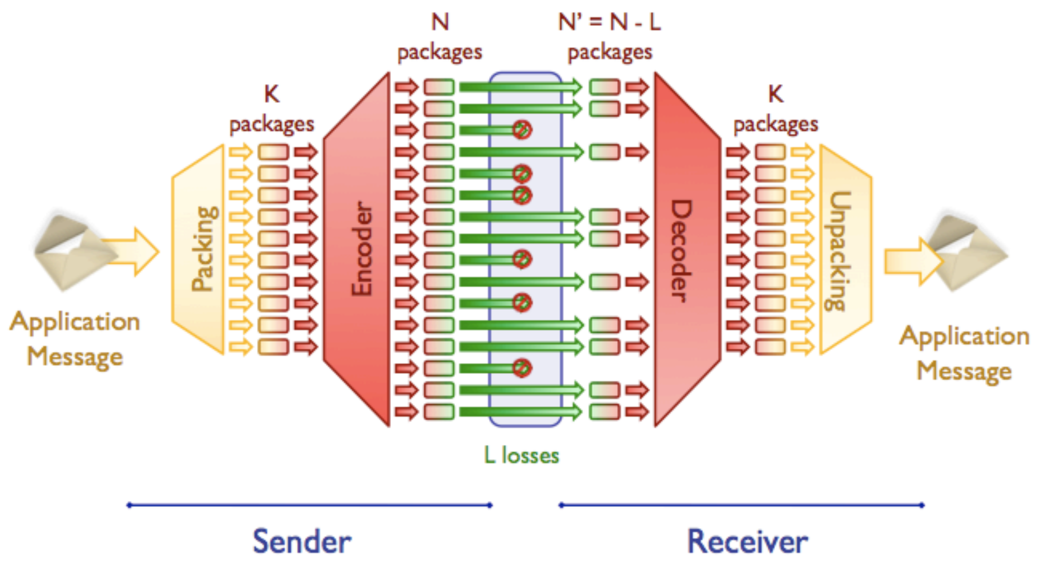


Figure 2.7: FEC-enabled interactions between a sender and a receiver

generic FEC technique is shown in Figure 2.7:

- the message passed by the application is packed in K blocks, with a given size σ;
- the K packets are passed to an encoder that, using on of the coding method discussed in Appendix A, generates the additional packets bringing to N the total number of packets to be delivered. The difference of the fan-in and fan-out of the encoder is a positive integer called redundancy degree, namely ρ, which is a measure of the redundancy that the encoder added to the data packets and almost equals the number of losses that the receiver can tolerate without contacting any other node for recovery:

$$\rho = N - K^{6} \tag{2.7}$$

- the N packets are exchanges through the network to reach destinations, however, some of them, named L, are dropped;

[6]Refer to Appendix A for further details on the recovery capability of coding techniques.

- the received N', equal to $N - L$, packets are used to feed the decoder and obtain the original K packets even if the delivery experienced some losses. However, such reconstruction is possible only the received packets are more or the same of the original data packets, i.e. $N' \geq K$, otherwise the reconstruction is not possible;
- the K packets are unpacked in order to obtain the original message.

Since the due of encoding the application message is placed in the multicaster, such recovery scheme is called **sender-initiated FEC**. Such techniques are characterized by the benefit of tunability, i.e. varying the pair of parameters (σ, ρ) allows providing a coherent relationship between performance overhead and reliability: more additional information added, the higher the performance overhead and the probability to correctly reconstruct the data packets. Such reliability strategy has two main drawbacks. On one hand, the detection of a lost message, *i.e.*, the one that cannot be reconstructed since the network dropped more packets than what FEC could tolerate, is possible only when the next message is received. On the other hand, redundancy degree is decided by the sender on the loss pattern experienced along the path of worst quality towards one of the destinations, so if only few paths exhibit heavy losses, then the source has to generate a large number of repair packets, even if the rest of destinations do not need it, leading to unwanted traffic on the network and high resource consumption[7].

[7]We refer to this issue as the "the boat unbalanced by the heaviest" problem.

2.2.1.5 Receiver-initiated FEC

The scope of **receiver-initiated FEC** is to combine the scalability of gossip-based protocols with the tunability of sender-based FEC: additional data are generated at the receiver side, rather than the sender side, from the received message and exchanged with randomly chosen receivers within the group. This can resolve the two problems affecting sender-based FEC. On one hand, the detection of dropped messages depends on the time to multicast additional information in the entire group rather than the inter-send time between consecutive messages. In fact, the recovery is based on the concept of region, which is the union of the nodes that belong to the same groups. Therefore, after selecting with whom it will gossip, each destination can aggregate in the FEC encoding all the messages exchanged in the groups that it has in common with the other destination. So gossiping is not triggered by the message inter-arrival time within a group, but within a region. On the other hand, the redundancy degree is not chosen in a centralized manner, resolving the problem of calibrating the redundancy degree on the worst case. Moreover, in case of no losses the strategy do not exhibit any additional overhead.

Receiver-initiated FEC was first introduced in the Slingshot protocol [71], aiming at recovering dropped messages in time proportional to the data rate in a single group. Further, an evolution of such strategy, named *Lateral Error Correction* (LEC), has been adopted in the Ricochet protocol [81], where recovery rate depends on the data rate at a node across several groups. In this latter protocol, a node belong to several distinct groups, and when it receives a message, it computes a repair packets as the XOR aggregation of the received message and $(r-1)$-previous ones. Then, it randomly chooses the destination of the repair packets and sends it. The destination can construct one of the aggregated messages by XOR the received repair packets and the $r-1$ messages available. The choice of which messages to aggregated depends on which groups the destination of the repair packet joined. LEC has been proved to be a scalable and timely strategy to reliably disseminate messages within a group of nodes, but, as all gossip-based strategies, it keeps to guarantee reliability only in a probabilistic manner. For this reason, Ricochet allows destinations to trigger retransmissions by sending NACKs to the multicaster of the dropped messages that local gossiping failed to recover.

2.2.1.6 Hybrid schemes

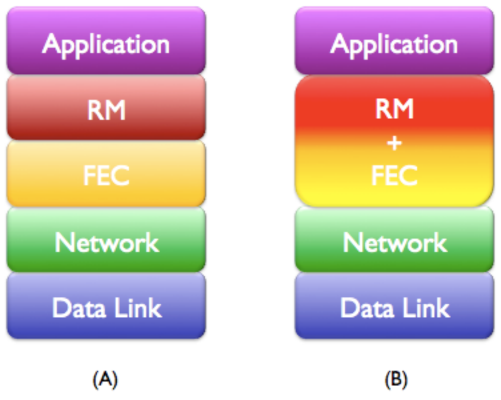


Figure 2.8: Placement of the FEC layer within the stack of protocols for Reliable Multicast: a) layered and b) integrated approach

As said at the beginning of this section, spacial and temporal redundancy can be combined in order to achieve both the strong reliability guarantees supported by the latter approach jointly with the timeliness features of the former one. In subsection 2.2.1.5, we have provided a practical example of a solution that combine the proactivity offered by

FEC and the reactivity of Gossiping in order to improve the timeliness and reliability. However, this is not the only solution to implement an hybrid scheme. During the years, there have been published several papers that illustrated several ways to introduce FEC into a retransmission-based reliable multicast, Figure 2.8 shows the main two generic approaches [84]. The first approach is referred as *layered FEC*: a FEC layer can be inserted below the protocol, namely RM layer, that realizes the reliable multicast, so that this layer has a more reliable view of the underlying network layer. When RM layer receive a message from the application layer, it can behave according one of the approaches explained in the context of temporal redundancy. However, before sending over the network, the message in encoded in order to generate additional information. At the receiver side, the received data are decoded in order to reconstruct the original data. The upper RM layer will receive the message in case of FEC success, otherwise, a notification of lost message, then it keeps behaving according to the chosen ARQ scheme. Such separation in two layers according to a "divide et impera" perspective simplifies the design of the hybrid protocol, in fact, the designer can focus on one problem (ARQ or FEC) per time. Although [85] proved a sensible reduction in network traffic by layered FEC, studies conducted in [84] have demonstrated that better improvements can be achieved when FEC and ARQ are integrated rather than layered, leading to a novel approach called *Integrated FEC*. In layered FEC, when a loss is detected, all the received packets will be discarded and data packets and new parity will reach the destination. In integrated FEC, if the received data is not enough to reconstruct the original message, new parity data are generated and sent to the destination, which keeps stored what previously received. This approach increases the probability to successfully reconstruct the original message after receiving a retransmission.

2.2.2 Loss Recovery In Application-Level Multicast

Loss recovery in ALM is significantly different from that in TLM [63]: in IP Multicast, the dissemination failure occurred in a destination (*i.e.*, its crash or the unsuccessful reception of a message) does not compromise the reliability of the data dissemination of other destinations. This is not true in an overlay network. In fact, messages are exchanged via piece-wise unicast connections among the members of the groups according to a certain overlay structure. The dissemination failure of a process can compromise the dissemination of all the other processes connected to the one that experienced the failure. Even if loss recovery in ALM is more challenging, it is also more flexible since processes have more means to easily cooperate to recover a dropped message.

The ongoing debate on message dissemination in ALM is to determine the right approach, in terms of performance and reliability, to organize the topology of the participants to a multicast session [67]. On one hand, there are **tree-based approaches** [68] that propose a structured organization of the nodes using a tree, where each node can implicitly define its parent, from which it receives the incoming messages, and children, to which it dispatch the incoming messages. On the other hand, there are **mesh-based approaches** [86] that expose a less structured organization letting each node to employ a swarming delivery mechanism to a certain subset of nodes. Mesh-based approaches are more resilient to process crashes and network disconnections since they do not have a rigid structure and can more easily adapt to changing conditions. However, mesh are more complex to built and maintain than trees since nodes needs to have knowledge of some of the other ones, which is challenging to obtain in Internet-scale systems. For this reasons, some protocols tries to combine the two approaches, *e.g.*, two recent practical example are the followings:

1. clustering nodes according to a proximity measure, where unstructured approach is adopted, and impose a tree-based structure among the clusters [87];
2. adopting a tree-based approach to deliver messages and trigger mesh-based recovery strategy to recover dropped messages [88].

Since there are these two opposing ways to organize processes within an ALM infrastructure, there are two possible delivery strategies: push-based, *i.e.*, new messages are delivered to other nodes whenever they are received, and pull-based, *i.e.*, identifiers of new messages are disseminated, so that the node that wants a given message has to request for its transmission. Usually trees adopts only push-based delivery, while meshes both. Performance comparison studies [86, 88] have proved that push delivery consistently exhibits superior performance over the pull delivery, but the latter one presents a more network-friendly behaviour by imposing lower traffic load.

In the last decades, several recovery approaches has been proposed to increase reliability properties of ALM solutions; some of them are specifically ideated for tree-base or mesh-based ALM and others can be applied in both approaches. In general, such strategies can be grossly grouped in proactive and reactive schemes. The former ones actively sends out redundant packets by using spatial redundancy, which will help processes to recover dropped packets without requiring any retransmissions. The latter ones adopts spatial

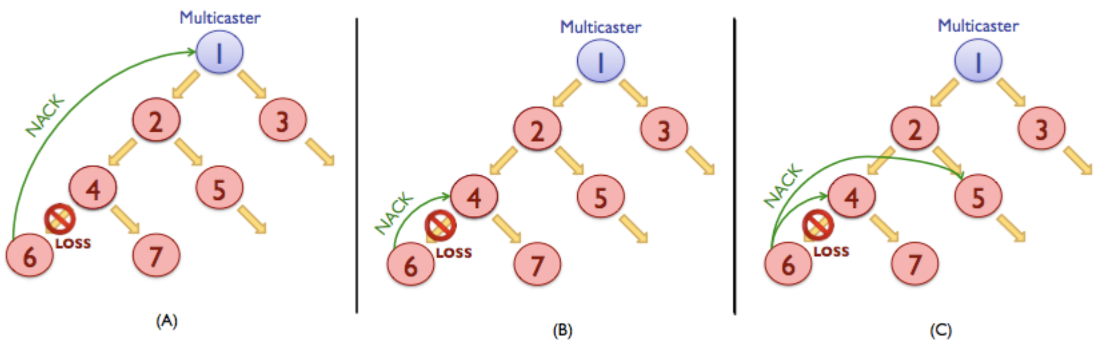


Figure 2.9: a) Sender-based, b) Parent-based and Multi-Parent ARQ Schemes

redundancy by allowing retransmissions of lost data in order to conduct recovery. As we have discussed for TLM, proactive approaches exhibit a low recovery latency due to the absence of retransmissions, but are complex to implement since the amount of redundancy needed to recover messages under all kind of failure load is hard to tune and cannot provide perfect reliability. On the other hand, reactive approaches are easy to implement and guarantee strong reliability properties, however, they present serious performance fluctuations when failure occurs. In the rest of this section, several recovery strategies for ALM are illustrated, pointing out their pros and cons.

2.2.2.1 ARQ Schemes over Overlay Networks

As discussed in subsection 2.2.1.1, **ARQ Schemes** belong to the category of reactive approaches: data sources usual assign a sequence_id to each message, while destinations can check the gap in the sequence_id of received messages to detect losses and request retransmissions. Depending on the receiver of the retransmission request, different ARQ schemes have been developed over overlay networks.

The most simple one comes directly from the literature of TLM recovery schemes, and consists of centralizing the retransmission responsibility in the multicaster [80], as illustrated in Figure 2.9A, for this reason it is called *sender-based retransmissions.* The strength of this approach is that the multicaster will surely be able to retransmit the dropped message. The weakness is that, when a process misses a message, also all the connected ones will lose it, so a burst of retransmission requests will reach the multicasting, causing the well-known NACK implosion problem. As we have seen for received-initiated TLM recovery, a possible solution is to delay the send of the retransmission request by a random time, but this would lead to higher recovery time. In addition, such scheme is not able to scale, so it is only suitable to small-scale ALM [63].

A solution to improve the scalability of retransmission-based recovery techniques is to distribute the retransmission responsibilities along the process organization [89], similarly to the hierarchical approach that has been presented in subsection 2.2.1.2. If processes are organized in a tree structure though unicast connections, a process can request the retransmission of a dropped message to its parent rather then to the multicast, as illustrated in Figure 2.9B. In this case, it is not certain that the contacted process may have the requested message. Therefore, if also the parent has missed the given message, it has to contact its own parent for retransmission (and so on, until the multicaster is contacted). When the message has been recovered, the process has to immediately forwards it to its children. This *Parent-based recovery* is scalable (there is no NACK implosion) and efficient (the recovery latency is not high, in fact, the time to deliver a message between parent and children is often low since processes select neighbors as parents). However, it presents a serious drawbacks called loss-correlation problem [63]: when a process misses a message, all its descendants will miss it, as a result, a request can be exchanged several times before the retransmission can be performed leading to high recovery latency at the descendants. In addition, this approach does not tolerant the failure of a link or a process that may disconnect a process. A workaround only to the latter issue is to allow processes to contact not only its parent but also its ancestors on the path to the source if a reply from the parent is not received.

To jointly handle the NACK implosion and the loss-correlation problem, a more smart solution is to use *multi parents* [90], *i.e.*, a process contacts its parent and its peers, as presented in Figure 2.9C. As a result, this approach exhibits multiple sources of retransmission, so the probability to fast recover dropped messages increases and a process can recover a message even if its parent has missed it. Multi-parent solution, however, does not deal with loss-correlation, since all the contacted processes have lost the same message if their parent have missed it, too. In addition, recovery may be high since the contacted "uncle" process may not be close. A better approach is to adopts a more generic *neighbor-based retransmissions*, where retransmission duties are distributed among any processes close to a given one[8] [63]. There are three main techniques to implement neighbor-based retransmission, which are the the focus of the next three subsections: 1) Lateral Error Recovery (LEC) is discussed in subsection 2.2.2.2, Cooperative Error Recovery (CER) is presented in subsection 2.2.2.3, while Gossiping is studied in subsection 2.2.2.4. In general, such techniques allows obtaining fast recovery and scalability, but they're are complex to implement. In

[8]The previous three strategies are mostly adopted when the ALM has a tree-base organization of its nodes, while this strategy is also strongly adopted in mesh-based ALM since it do not require a structured organization.

fact. the selection of the which process to contact is not trivial since it requires the global knowledge of all the participant to a given group, which is improbable to obtain in case of large-scale systems over wide-area networks.

2.2.2.2 Lateral Error Recovery (LER)

Lateral Error Recovery (LER) [91] comes from the consideration that traditional "vertical" recovery strategies, *i.e.*, such as the previous ones where request for retransmissions are propagated along the path to the source, are not suitable when messages need to be timely recovered. Therefore, as suggested by its name, LEC adopts a novel "lateral" perspective to realize faster recovery, *i.e.*, all the members of a group are randomly grouped in ω distinct planes, where multicast trees are independently formed. The root of a tree in a given plane is called plane source and it is directly connected to the multicaster. Each process select as neighbors one process per each plane that is different than its own. When a process experiences a message loss, it request a retransmission to its neighbor that exhibit lower recovery latency. If this process does not have the requested message, one of the other neighbors are contacted, while if none of the neighbors contains the given message, then the source plane and the multicasted are contacted, as illustrated in Figure 2.10. Since processes are randomly chosen to belong to a certain plane and messages are delivered along different trees, the loss correlation across the planes are low. This enforce the efficiency of the protocol since there is a high probability that the first contacted neighbor holds the requested message. On the other hand, since there is no centralized organization of the

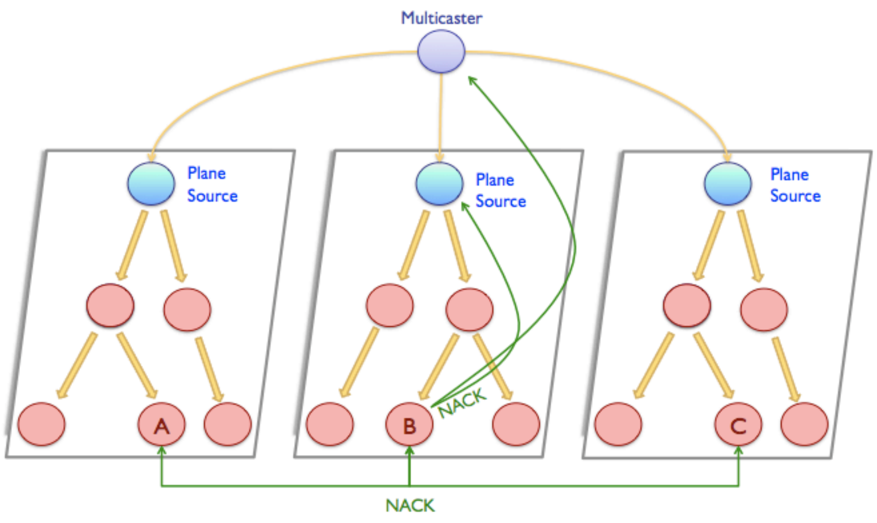


Figure 2.10: Organization in Tree Planes of LER

retransmissions, scalability is obtained. The main weakness is that it requires high measurements and computation overhead in order to create the planes and select the neighbors of a process.

2.2.2.3 Cooperative Error Recovery (CER)

Cooperative Error Recovery (CER) [92] is another methods to select neighbors for efficient message recovery. Processes are clustered in the so-called *minimum-loss correlation* (MLC) groups, whose members are characterized by a negligible loss correlation, *i.e.*, it is unlikely that, if a process experiences a message loss, the members of it MLC group lost its same message. MLC groups are constructed in two steps. First, processes exchanges neighbor informations along the multicast tree, so a process will know about a medium-sized subset of other processes. Then, the process calculates its loss correlation with all the known processes, and consider its neighbors the ones that exhibit minimized correlation. If a process detects a loss, it will inform its children, and then, it will contact its closest neighbor in order to recover the message (it will keeps on requesting if the contacted process does not have the requested message). If a message has detected a large number of consecutive lost messages, it will deduce that there is a parent failure or a congestion, and start a rejoin process.

2.2.2.4 Gossiping in Overlay Networks

The idea of adopting the epidemiology theory to realize reliable multicast, which has been discussed in sub-section 2.2.1.3, has been adopted in several ALM solutions. The success of these gossip-based ALM protocols is motivated by their inherent scalability and easy deployment over Internet or ad hoc networks, but even more by their resiliency to failure [93]. In fact, it is easy to adjust the parameters of the gossiping algorithm in order to achieve high reliability despite several kind of failures, *e.g.*, packet losses, process crashes or network partitioning. As said in 2.2.1.3, a node store a received message in a buffer with a capacity b, and forwards the message a limited number of times t to a randomly-selected set of processes of size f. Many variants of epidemic algorithms exist, characterized by values of b, t and f, but they can be summarized in one of three following approaches [94]:

- *Push Approach*: received messages are forwarded to random selected neighbors;
- *Pull Approach*: nodes periodically query random selected neighbors asking for a list of recently-received messages by sending a gossip message. If a new information (i.e., a lost message) is detected in the received gossip message, then a transmission is requested.

- *Push/Pull Approach*: When a node receives a message for the first time, it forwards to some random neighbors only its identifier, and not its content. If one of the receivers does not have such message, then it makes an explicit pull request.

In push approaches, data dissemination and recovery phases are integrated, and nodes are organized according to an unstructured mesh-based overlay, equipped with a sampling service that provide to each node a, partial or global, view of its neighbors [95]. Unlike these approaches, the pull-based ones deal only with the recovery of lost information, and may be associated with any reliable broadcast mechanism. So, recent gossiping solutions has also been integrated with tree-based delivery in order to have a small message complexity in the steady-state [96].

The main drawback of such solutions is that delivery exhibit only probabilistic reliability assurance, even if manipulating the protocols parameters the probability to receive messages is high. Push have proactive nature but is not very network-friendly. To reduce the need of global knowledge of the nodes composing the systems and adapt the network overhead to the experiences loss pattern, in [97] a *hierarchical gossiping* is presented, where nodes are clustered in overlapping groups based on topological informations and gossiping is performed locally at each group. In the recent [98], a different adaptation technique, known as *gravitational gossiping*, is presented in order to adapt gossip settings to meet different dissemination rate targets as network conditions change by clustering nodes not only according to distance but also to interest to certain information.

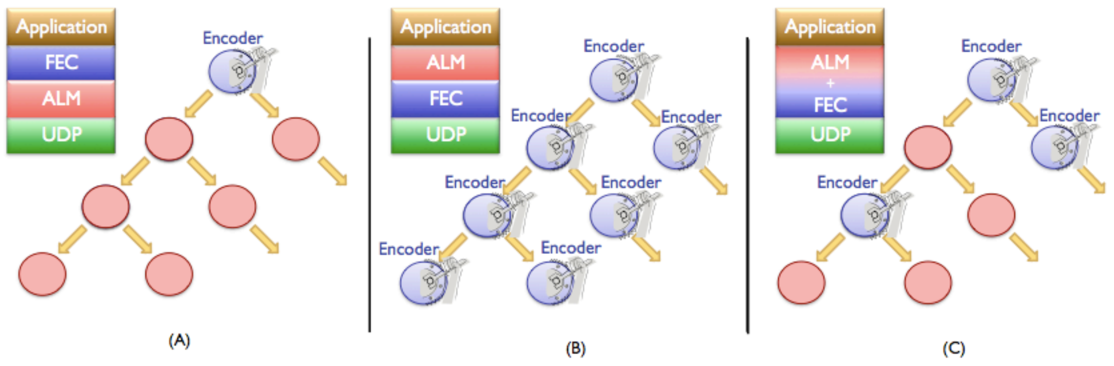


Figure 2.11: Three possible FEC-enhanced methods for loss-tolerant ALM: a) End-to-End FEC, b) Link-by-Link FEC and c) Selective Network-Embedded FEC

2.2.2.5 FEC Approaches over Overlay Networks

As previously illustrated for TLM, also some ALM solutions, all of them based on a tree organization, adopts Forward Error Correction to tolerate message losses. There are three different approaches, which differ each other with respect of where the encoding of additional data is performed, as presented in Figure 2.7:

- *End-to-End FEC* [99]: encoding of the messages to be delivered through the overlay network is performed only by the multicaster, as illustrated in Figure 2.11a, while all the destinations decode the received packets in order to obtain the original message even if the network was affected by losses. Such approach can be abstracted as follows: a FEC layer is placed between the ALM protocol and the business application, so messages generated by the application are encoded before been passed to the ALM, which will carry on the dissemination, while the packets received by the ALM will be decoded in order to be delivered to the application. This method is easy to implement, due to such layer separation, while it is strongly affected by the issue that we referred as "the boat unbalanced by the heaviest" in sub-section 2.2.1.4. In fact, the redundancy degree is chosen only by the multicaster and depends on the loss pattern of the worst-case path from the multicaster to one of the destinations. Since overlay networks deliver a message by passing though several overlay links and processes, as shown in Figure 2.1, it is likely that destinations will experience highly-heterogeneous loss pattern along the path from the destination: the ones closer to the multicasting will need a redundancy much lower than the ones that are more far away. So, in order to assure message delivery, the multicaster has to impose an overwhelming redundancy degree that may overload the nodes that needed less redundancy and/or cause serious congestion in parts of the network.

- *Link-by-Link FEC* [100]: this solution adopts an opposite perspective than the previous one. Specifically, every node in the overlay network perform encoding and decoding (the multicaster only executes encoding), in order to protect the delivery on each channel from message losses. As shown in Figure 2.11b, it is formalized by putting the FEC layer between the ALM and the UDP protocols, so every delivery operation invoked by the ALM will trigger an encoding, and every receive a decoding. This method is more flexible since the redundancy degree is chosen only with respect to the quality of an overlay link between two nodes. This resolves the "the boat unbalanced by the heaviest", but also causes strong degradations in performance due to the continuous execution of the two coding operations at every overlay node.

- *Selective Network-Embedded FEC* [101]: This approach represents the mean between the previous two methods: only a subset of nodes, which includes the multicaster, is able to perform encoding, while only destinations can decode the received packets. This allows to optimize the need of a flexible setting of the redundancy degree in order to avoid excessive over-provisioning without worsening the achievable reliability degree in case of message losses. Such protocol is usually implemented by integrating FEC and ALM layers, and delivery is realized throw a tree-based approach since it provides direct control over the path followed by messages. Its main drawback is that all the techniques proposed in literature requires global knowledge of system topology in order to resolve the problem of choosing where to perform encoding (*median problem*), or they are affected by scalability issues limiting its applicability to small-scale system.

2.2.2.6 Path Redundancy

Considering that several prior studies have demonstrated current Internet exhibiting redundant connections at AS-level [102, 103, 104], an overlay node has more than on path to reach its destination. Over the past years, several papers has been published, such as [6], that take advantage of the implicit Internet redundancy and have proposed routing optimization to tolerate losses imposed by congestion, malfunctioning networking devices or BGP misconfigurations. Such optimizations consist either probing to find the single path that exhibit best properties in terms of message losses and connection availability (*Best-Path*), either sending data redundantly over several distinct paths (*Redundant-Paths*). In our opinion the first one is more a loss-avoidance rather than a loss-tolerance technique, in fact, selecting the best path allows to reduce the probability to lose messages but does not provide any mean to guarantee that a message is correctly delivered. For this reason we have focused our attention on **Path Redundancy**, which can be applied in a reactive and in a proactive manner and is a powerful mean to circumvent lossy links or crashed elements. In the first case, the overlay node holds a primary path to forward messages to an other node and back-up paths to use in case the primary has experience a loss [105]. On the other hand, the possibility to use several redundant paths at the same time to deliver to a given node several copies of a given message is very promising to have stable performance even in case of message losses. In fact, no retransmissions are needed and the delivery time presents a known upper bound, which is equal to the latency of the worst path. The achievement of strong reliability guarantees is obtained only if path diversity among the disjoint paths is verified [20], *i.e.*, if a message on a certain path is dropped, the replicas exchanged on the other paths are certainly delivered to the destination. However, such point of strength do not come for free, in fact, such protocols pay an high price in terms of traffic load due to the adopted redundancy. In fact, there are different approached to use the redundant paths, not just the multicopy transmission [106] that we have introduced in the previous statements, that aim to reduce the traffic overhead paying a reduction in the achieved reliability:

- partition the information data in stripes [107] and deliver then through some paths while their copies in the other ones,
- segment the application data in blocks, encoding them to obtain redundant blocks [108] and disseminate the original and redundant blocks through the redundant paths;
- creating redundant data operating linear combinations of the information data [109],

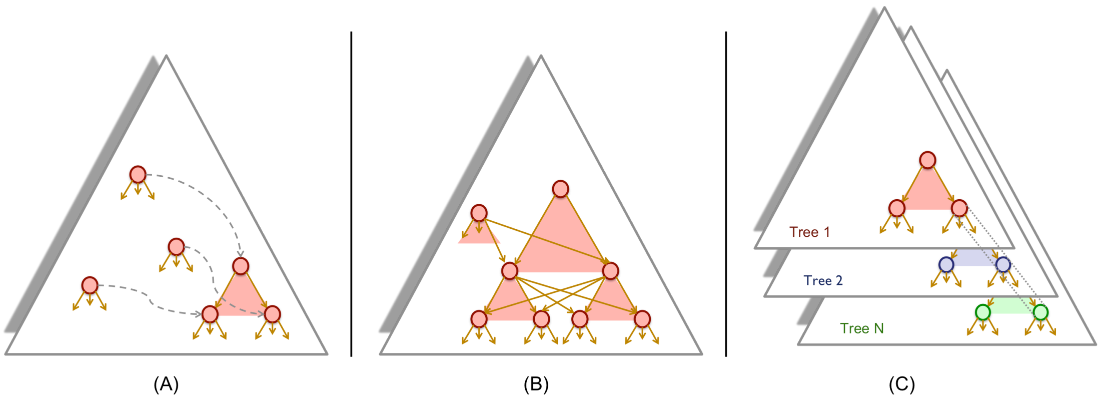


Figure 2.12: Approaches based on path redundancy: a) cross-link, b) in-tree, and c) multi-trees

and distribute the obtained original and redundant data over several paths.

Mesh routing offer the simplest way to implement path redundancy by randomly choosing more that one neighbor node to communicate with [110]. A more sophisticated strategy uses swarning content delivery by combining push content reporting and pull content requesting [111]: periodically each process forwards to all of its children a list of the newly received messages (push phase), so that one of its process can request the transmission of a specific message (pull phase). On the other hand, such strategy is also widely used in tree-based ALM. In fact, there are three alternative approaches that have been presented in the last years [112, 113], as shown in Figure 2.12:

- cross-link [114], *i.e.*, connecting random peers via extra cross-cutting links. A process forwards the incoming messages not only to its children in the tree, but over its extra links to random peers.
- in-tree [115], *i.e.*, extra- links among different "families of processes are established. In a tree-based organization, processes depends on their parent to receive a message,

and this dependency on a single process is the main cause of the low reliability in case of message losses. In-tree redundancy give to a process the possibility to have an additional parent from which it can receive messages.

- multi-trees [19], *i.e.*, multiple interior-node disjoint trees, namely forest, are formed among the members of a group over which

While all of them have been demonstrated to improve the resiliency of the multicast service, the multiple-tree exhibits a sensible reduction of the unrecovered messages and better guarantees to satisfy stringent real-time deadlines [116].

2.3 Discussion

The previous sections provided a brief overview of the loss tolerant techniques that have been proposed during the years by the international research community, and Figure 2.13 presents a taxonomy that summarizes such overview. Facing such a huge number of proposed techniques, we need to answer the following key research questions: what the research on reliable multicasting has achieved so far, what is still missing and how the requirements expressed in subsection 1.3.1 can be obtained by current loss-tolerant publish/subscribe services for the class of systems presented in section 1.1. The following two subsections respectively study the available loss-tolerant strategies applied in TLM and ALM solutions and discuss their quality in terms of scalability, reliability and timeliness. Based on the considerations introduced in these two subsection, we conclude the section by highlighting the limitations of the loss-tolerant solutions adopted in reliable publish/subscribe services that prejudice the applicability of these available approaches to data dissemination in LCCIs.

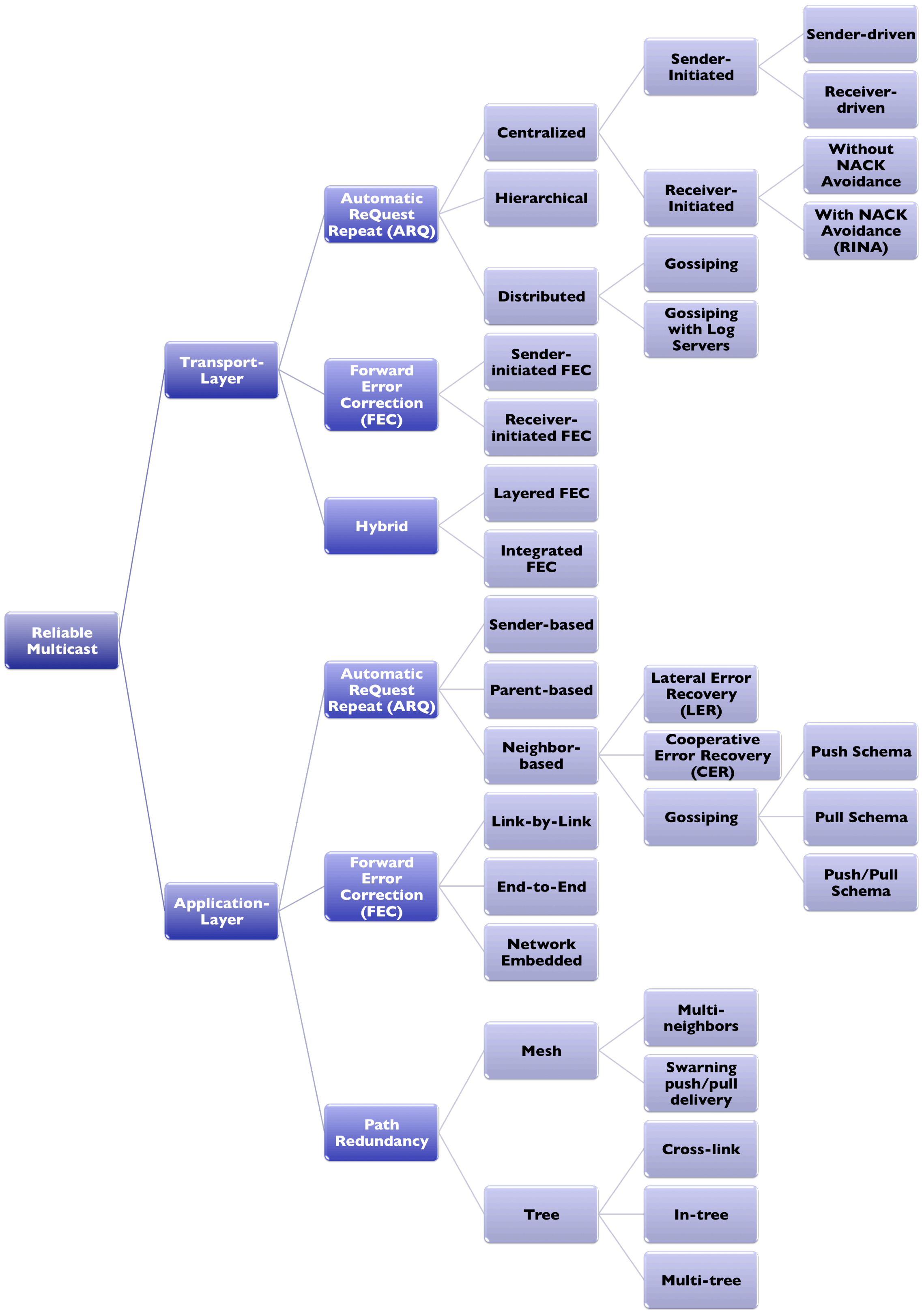


Figure 2.13: Classification of all the approaches available in literature to tolerate dropped messages

2.3.1 Efficiency of TLM approaches

As we said at the beginning of subsection 2.2.1, ARQ schemes are based on temporal redundancy that allows high degree of reliability since a given message is retransmitted until it is successfully received. Therefore, such schemes are able to tolerate sporadic message losses that may affect the network, but some of them are unable to deal with crashes. Moreover, as clearly stated by Equation 2.4, they provide low timeliness assurance since the time to deliver a message when losses occur strongly depends on the network conditions, which are known to be unpredicatable. Specifically, centralized approaches are affected by several implosion problems that negatively affect the scalability exhibited by the approaches. Moreover, if the multicaster fails, the destinations are unable to recover dropped messages. The hierarchical approach to loss tolerance allows improving the scalability and reliability degree of the overall multicast service. In fact, implosion problems no more represent an issue, and the multicaster is not a single point of failure since its failure do not compromise the recovery of dropped messages. However, nodes at the lower tier of the hierarchical organization are affected by higher recovery latency for the loss correlation problem. Such issue is treated by distributing the recovery responsibilities using a gossiping approach, which allows reducing the number of needed retransmissions increasing the timeliness by contacting more than one process to recover a lost message. To achieve high timeliness degree the most powerful approach is to apply spatial redundancy by adopting FEC techniques, as described in Equation 2.6. However, sender-initiated FEC is affected by the "the boat unbalanced by the heaviest" problem that drastically reduce the scale of destinations that

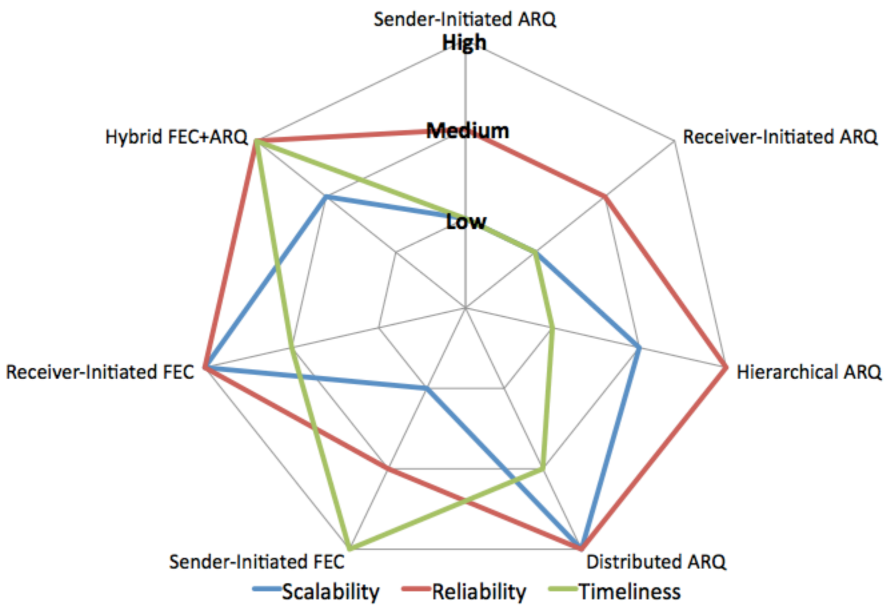


Figure 2.14: Analysis of the quality of loss-tolerance for TLM in terms of scalability, reliability and timeliness

the approach can handle in an heterogeneous network scenario. Moreover, FEC techniques do not exhibit a strong reliability degree since, if the decoding fails (*i.e.*, the number of lost messages is greater than the applied redundancy) destinations do not have a way to recover dropped messages. Redundancy degree is challenging to optimally set since the loss pattern is unknown a priori. Receiver-initiated FEC has been demonstrated in [81] to exhibit strong scalability and reliability degrees. Also, combined FEC and ARQ solutions allows achieving good timeliness and reliability assurances, however, scalability is not high since the coding duty is centralized at the multicasting, causing the 'the boat unbalanced by the heaviest" problem, even if the consequences of such problem are mitigated by the retransmissions unlike the case of sender-initiated FEC. Such considerations are graphically presented in the radar chart shown in Figure 2.14.

2.3.2 Efficiency of ALM approaches

The consideration about ARQ schemes that we have made in the context of TLM in the previous section remain valid also in the context of ALM. In fact, reliability is successfully achieved by using retransmissions at the expenses of timeliness (even if using a distributed approach, *e.g.*, LEC, CER and Gossiping, allows to reduce the performance overhead imposed to recover a lost message). With respect to the scalability, sender-based exhibit the worst guarantees since recovery is centralized at the multicaster, while the other approaches improves the overall scalability by distributing the recovery duties among the nodes participating to the multicast session. Specifically, distributed approaches obtain the high scalability score with the exception of LER, which require a details knowledge of the overall system topology, which is impractical for large scale systems. On the other hand, FEC-based techniques realize a timely message delivery even in case of losses paying a reduction of reliability (End-to-End FEC has the worst reliability degree since its is more challenging to opportunely tune the FEC redundancy and the probability to have to deal with the "the boat unbalanced by the heaviest" problem is higher). Concerning scalability, the three FEC solutions exhibit divergent behaviors: the link-by-Link FEC has the worst performance at the lower layer of the disseminating tree[9], End-to-End FEC do not suffer of this issue but the centralization of the FEC at the root of the tree limits the number of nodes that can be handled, last Selective Network Embedded FEC does not suffer of any of this problems. On the other hand, Path Redundancy has high timeliness properties when it is used in a

[9]If we fix the layers of the multicast tree, a large number of nodes will be placed at the same layer, even if they are not all close to each other. This imply that the parent of such mass of node will have to impose a high redundancy degree, with a consequent degradation in performance.

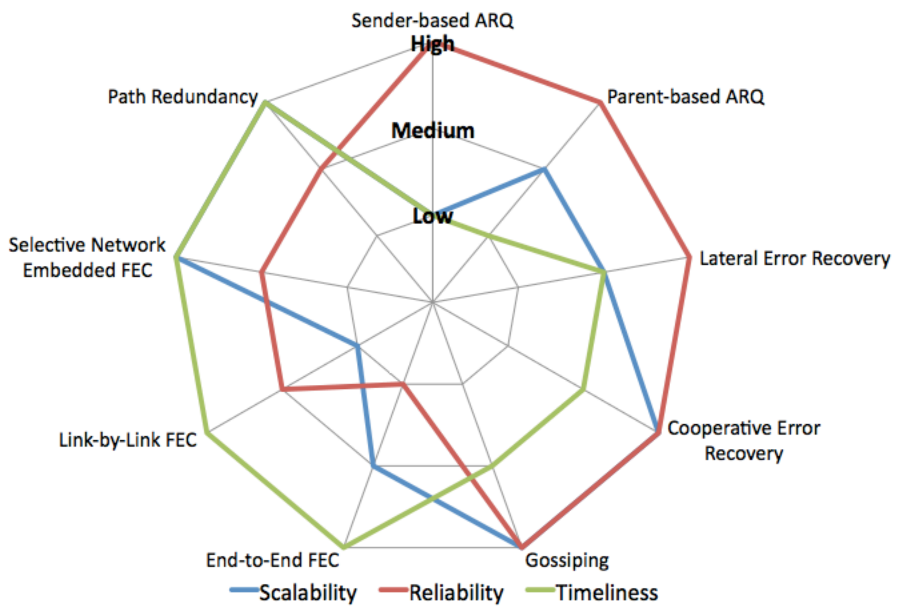
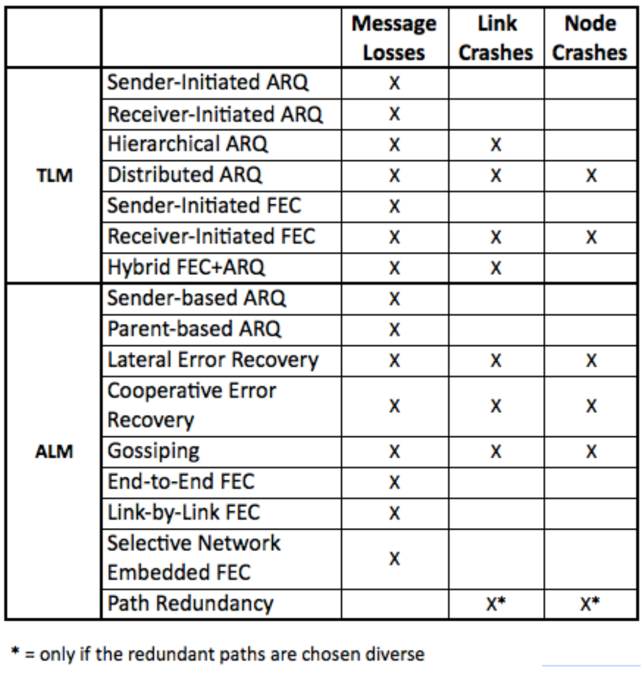


Figure 2.15: Analysis of the quality of loss-tolerance for ALM in terms of scalability, reliability and timeliness

proactive manner, even if the reliability strongly depends on the diversity of the used paths. This techniques can manage large number of nodes without any problem. Such considerations are graphically presented in the radar chart shown in Figure 2.15, which clearly shows that retransmission-based schemes exhibit the highest reliability degree at the expenses of timeliness, while proactive techniques, *e.g.*, FEC and path redundancy, exhibit an opposing behavior since timeliness is enforced paying a reduction in the reliability of the message dissemination. If we look at the possible failures tolerated by retransmission approaches, we can notice that Sender-based and Parent-based ARQ are vulnerable to link and node crashes due to the static dependency of each node to the sender (in the first approach) and to the parents (in the second one) for recovering lost messages, while distributed solutions can easily handle also such crashes. On the other hand, the illustrated FEC techniques are tailored only on sporadic losses imposed by the network and are ineffective in case of crashes, while path redundancy can handle more easily such failures by circumventing the crashed element but is less effective in case of message losses if they are correlated among the redundant paths.

Table 2.1: Study of the failures tolerated by the reviewed approaches



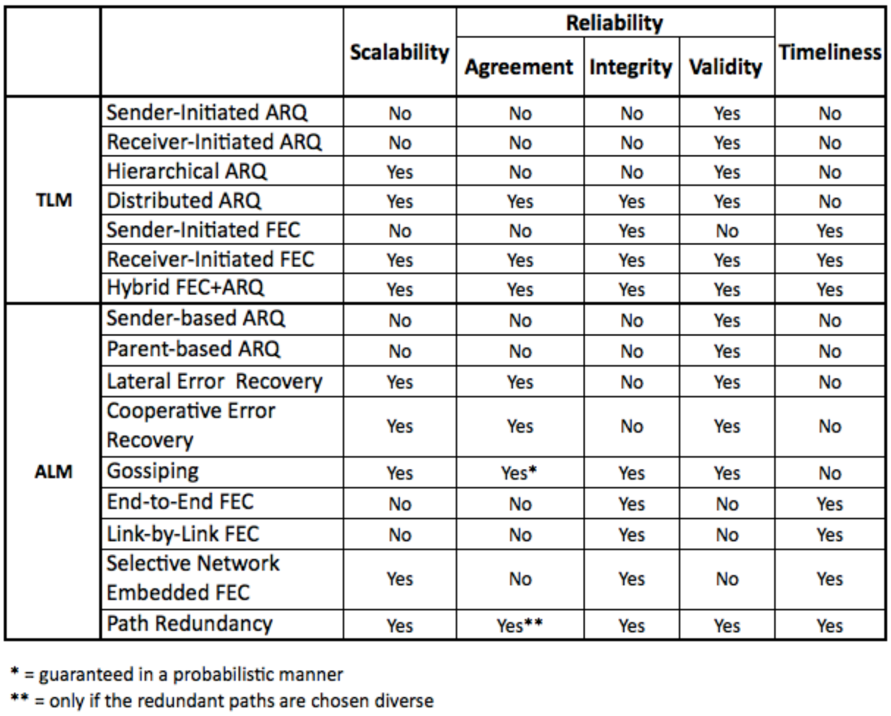
		Message Losses	Link Crashes	Node Crashes
	Sender-Initiated ARQ	X		
	Receiver-Initiated ARQ	X		
	Hierarchical ARQ	X	X	
TLM	Distributed ARQ	X	X	X
	Sender-Initiated FEC	X		
	Receiver-Initiated FEC	X	X	X
	Hybrid FEC+ARQ	X	X	
	Sender-based ARQ	X		
	Parent-based ARQ	X		
	Lateral Error Recovery	X	X	X
	Cooperative Error Recovery	X	X	X
ALM	Gossiping	X	X	X
	End-to-End FEC	X		
	Link-by-Link FEC	X		
	Selective Network Embedded FEC	X		
	Path Redundancy		X*	X*

* = only if the redundant paths are chosen diverse

2.3.3 Loss-tolerance of current reliable publish/subscribe services

The study of the failures managed by the approaches presented in subsections 2.2.1 and 2.2.2 are reassumed in Table 2.1. While, the considerations on the provided non-functional properties, *i.e.*, scalability, reliability and timeliness, made in the previous subsection are schematically presented in Table 2.2. Based on the consideration summarized in these two tables, we aim to discuss the suitability of available reliable publish/subscribe services for the reliable and timely event dissemination in the context of LCCIs.

Table 2.2: Scalability, Reliability and Timeliness analysis of the reviewed approaches



		Scalability	Reliability			Timeliness
			Agreement	Integrity	Validity	
TLM	Sender-Initiated ARQ	No	No	No	Yes	No
	Receiver-Initiated ARQ	No	No	No	Yes	No
	Hierarchical ARQ	Yes	No	No	Yes	No
	Distributed ARQ	Yes	Yes	Yes	Yes	No
	Sender-Initiated FEC	No	No	Yes	No	Yes
	Receiver-Initiated FEC	Yes	Yes	Yes	Yes	Yes
	Hybrid FEC+ARQ	Yes	Yes	Yes	Yes	Yes
ALM	Sender-based ARQ	No	No	No	Yes	No
	Parent-based ARQ	No	No	No	Yes	No
	Lateral Error Recovery	Yes	Yes	No	Yes	No
	Cooperative Error Recovery	Yes	Yes	No	Yes	No
	Gossiping	Yes	Yes*	Yes	Yes	No
	End-to-End FEC	No	No	Yes	No	Yes
	Link-by-Link FEC	No	No	Yes	No	Yes
	Selective Network Embedded FEC	Yes	No	Yes	No	Yes
	Path Redundancy	Yes	Yes**	Yes	Yes	Yes

* = guaranteed in a probabilistic manner
** = only if the redundant paths are chosen diverse

As we have said in subsection 1.3.3, very few publish/subscribe services pay attention to tolerate losses imposed by the network behavior or link and node crashes. The ones that exhibit loss-tolerance capabilities adopt retransmissions either in a IP Multicast-based publish/subscribe solutions such as DDS, either in Overlay Multicast ones such as XNET [117]. However, as we have previously seen, such approaches are not suitable for LCCIs since timeliness is not guaranteed and also the exhibited scalability is not suitable for such large scale systems (even in case of IP Multicast deployed all over Internet). On the other hand, also gossiping approaches have been investigated, such as in [16], however, even if scalability is strongly improved, timeliness is still not provided. Recently, FEC techniques are started to been applied for event dissemination, such as in [118], but only for scenarios where IP

Multicast is enabled. Therefore, the available solutions are not suitable to address the challenges of LCCIs expressed in subsection 1.1.3. Another drawback in current approaches is that only one reliability strategy is adopted all over the dissemination strategy. Even if they use node clustering to be more network-friendly, they do not consider the possibility to select the reliability strategy depending on the specific conditions experienced by each group of nodes. LCCIs are characterized by federating heterogeneous systems, each one with a network infrastructure that present a certain behavior, *e.g.*, network with no losses or networks with string loss patterns. The perspective of "one solution fits all" is not optimal for LCCIs since does not deal with their intrinsic network heterogeneity.

Chapter 3

Reliable and Timely Data Dissemination over Internet

LCCIs adopt a federated architecture similar to the one that is currently exhibited by Internet. Based on this observation, we propose an innovative organization of an effective data delivery infrastructure for LCCI that is called TODAI (acronym for Two-tier Organization for Data Dissemination Infrastructure): nodes that reside on the same routing domain are clustered together, and different clusters are interconnected according to a super-peer topology. Although such organization strongly enforces scalability, it is affected by several kind of failures that can compromise the correct message dissemination. Therefore, the problem of guaranteeing reliable and timely event distribution is treated by applying a "divide et impera" approach by breaking down the overall problem into sub-problems where only a certain kind of failures can occur. So, we have separately investigated each sub-problem and proposed the best technique to tolerate the particular class of failure that characterizes each sub-problem. Specifically, the following solutions have been applied in TODAI:

1. *replication is used to tolerate node crashes without leading to disconnections among clusters;*
2. *a series of proactive techniques is adopted in order to jointly provide timeliness and reliability in the event dissemination:*
   - *(a) distributed FEC and multiple-tree dissemination are teamed up in order to handle link crashes and severe loss patterns that characterize the communications among different clusters;*
   - *(b) layered FEC is used for communications among nodes within a certain cluster with the scope of guaranteeing reliable data distribution and adapting the traffic load to the needs of each destination;*
3. *proactive techniques have been accompanied by a push-based gossip recovery strategy in order to improve the levels of reliability achieved with the previous proactive techniques.*

3.1 Two-tier Organization for Data Dissemination Infrastructure (TODAI)

LCCIs impose several tough requirements on the adopted data distribution infrastructure, as we have described in subsection 1.1.3. Specifically, the most challenging one is to assure a timely dissemination of critical informations to all the geographically-distributed systems. As written by Su Tze, the great military expert of ancient China, in his book titled "The art of the war", we have to beforehand know our enemy in order to win a battle. For LCCIs, the enemy is represented by Internet, which has been demonstrated to represent an hostile environment for an efficient data delivery due to its faulty bahaviour. For this reason, in subsection 3.1.1, we discuss about Internet and its topology. Based on the lessons learned from research and practice on how implement effective routing in Internet, an innovative architecture for Internet-scale publish/subscribe services is illustrated in subsection 3.1.2. Such architecture, named **TODAI** (*T*wo-tier *O*rganization for *DA*ta dissemination *I*nfrastructure), adopts a two tiers organization according to a super-peer topology: (*i*) nodes are grouped in clusters based on topology informations, while (*ii*) clusters are interconnected using an overlay tree-based network, as presented in subsection 3.1.3. We conclude this section by providing in subsection 3.1.4 an overview that explains how events are spread among the different geographically-distributed nodes.

3.1.1 Internet Topology

The Internet is by definition a meta-network (as also described by its name, which is a short form of the compound word "inter-networking"): a constantly changing collection

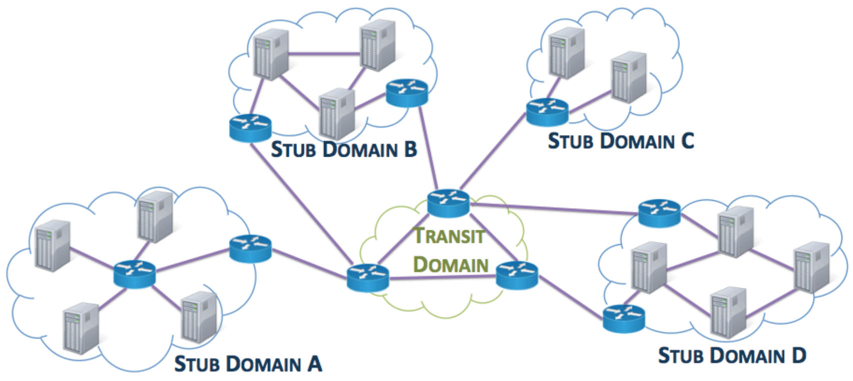


Figure 3.1: Model of Internet as a composition of Stub and Transit Domains

of thousands of intercommunicating individual networks. Therefore, the Internet's architecture is naturally modeled as a composition of several interconnected *Routing Domains*, each one shares common administration control and routing protocol [119], as illustrated in Figure 3.1. Such domains exhibit a hierarchical topological organization consisting of two abstraction levels. On one hand, there are the so-called *Stub Domains*, *i.e.*, routing domains that contain nodes communicating with each other using the inside local addresses. Practical examples of these domains consist of *Local Area Networks* (LANs) or *Autonomous Systems* (AS). The key feature of stub domains is that they are managed by a central organization. So, policies to assure Quality-of-Service (QoS) constraints in the data dissemination may be applied and IP Multicast may be made available to all the communications among nodes that reside on a given stub domain. On the other hand, *Transit Domains* are in charge of efficiently interconnecting several stub domains and to form the network backbone[1]. Due to a lack of a central management reference and traffic orchestrator, transit domains are

[1]For simplicity, in Figure 3.1 we have represented only one transit domain, but currently communications over Internet may traverse several ones when moving from one stub domain to another one.

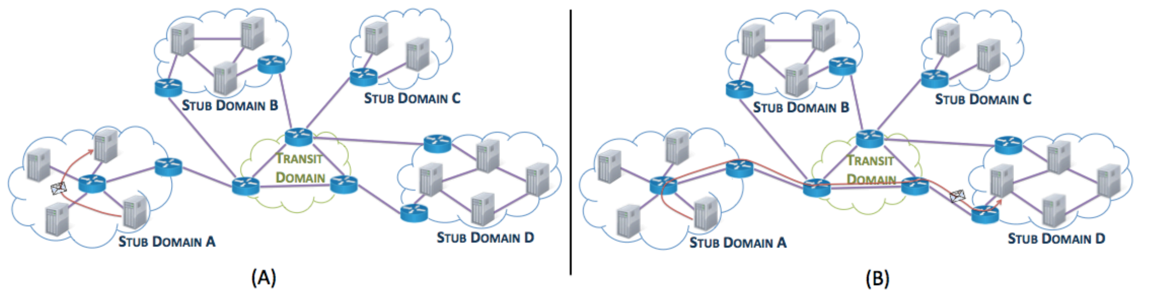


Figure 3.2: Event dissemination within an LCCI: (A) within a stub domain and (B) between two distinct stub domains

affected by several failures that may compromise the effectiveness and resiliency of the message forwarding. Although important technical progress [120] has been made to address this, more work needs to be done to achieve trustworthy QoS guarantees in Internet. In addition, as clearly expressed in section 2.1, communication among stub domains cannot be conveyed by IP Multicast since the routers in the transit domains do not provide it. Even if connectivity among routers supporting IP Multicast can be provided using point-to-point IP encapsulated tunnels [121], such solution exhibits severe reliability limitations,*i.e.*, it strongly results vulnerable to the failures of the routers at the end and along the tunnel, and maintainability issues, *i.e.*, the tunnel needs to be manually re-established by human operators every time a failure occur.

3.1.2 Innovative Internet-scale Publish/Subscribe Architecture

As introduced at the end of subsection 1.1.1, an LCCI represents a wide-area federation of several heterogeneous systems, each one managed by a certain organization, but without a centralized organization that has the capacity of controlling the overall LCCI. It is reasonable to assume that all the nodes belonging to one of these systems composing an LCCIs

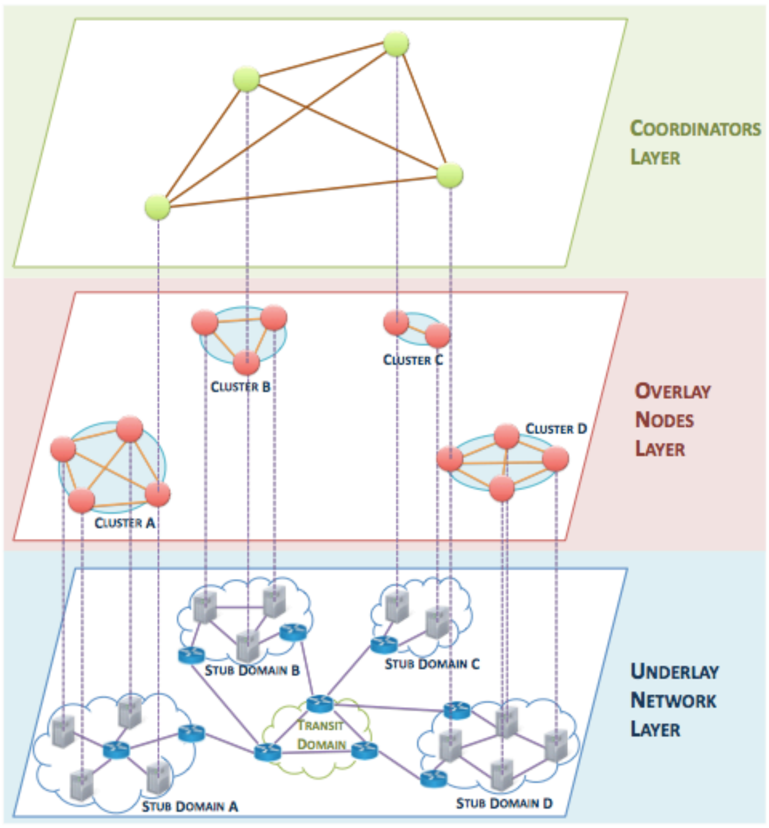


Figure 3.3: Layered organization of the data dissemination within an LCCI

reside on a given routing domain. Therefore, the issue of disseminating information within an LCCI consists of two distinct issues: 1) how to distribute information towards nodes that reside of the same stub domain, and 2) how to spread information among nodes placed on different stub domains. The most feasible approach to deal with such issues is to adopt the same solution successfully used for years in Internet[2]: organizing the dissemination infrastructure according to a two-tiers architecture, which is illustrated in Figure 3.3. Specifically, all the nodes that reside on the same stub routing domain are clustered together, forming

[2]Routing in Internet is segmenting as routing within an autonomous system by using the so-called *Interior-Gateway Protocol* (IGP), and among different autonomous systems by using the so-called *Border Gateway Protocol* (BGP) [122].

the first tier of the architecture. The definition of the cluster can be statically established by embedding in the application code of each node the identity of all the other nodes that form the cluster, or nodes can autonomously detect the other peers using a multicast beaconing mechanism. In fact, since nodes in the same cluster are placed in the same stub routing domain, where IP Multicast is available, a node can multicast a beaconing message and can infer as participants the nodes that are able to listen such message. IP Multicast can be used as a mean to convey data exchanged within a cluster so to achieve efficiency both in terms of judicious use of network bandwidth and scalable support to a large number of participants to a communication. However, if data need to reach a node belonging to a distinct cluster, cluster needs to be interconnected using Internet as networking infrastructure. Since transport-level multicasting is not feasible in such scenario, application-level multicasting represents the only viable solution. The scalable manner to interconnect different clusters is not to allow each node to communicate with nodes in other cluster, but to delegate the inter-cluster communication only to certain nodes called *coordinators*. In fact, the second tier of the architecture illustrated in Figure 3.3 is made of a peer-to-peer application-level multicast that interconnect the coordinators in each cluster. *Super-Peer architectures* [123] are known to provide a scalable message dissemination in the context of the Internet-scale application-level multicast [124]. Let consider Figure 3.4 where node 1 wants to disseminate a message to all the other nodes. In the left part of the figure, the nodes are architected according to a pure peer-to-peer topology and the tree built among all the nodes is made of several links among nodes placed in different stub domains (*i.e.*, six links in total, as shown by the red lines in the upper left corner of Figure 3.4). On the

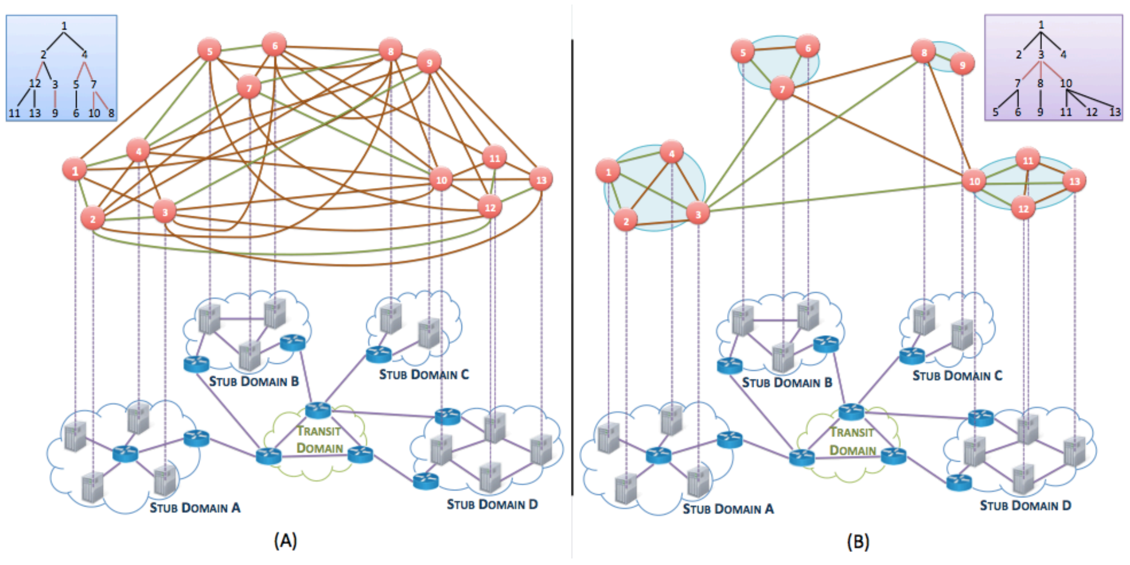


Figure 3.4: Comparison of (A) a pure peer-to-peer and (B) a super-peer topology

other hand, in the right part of the figure nodes are connected using a super-peer topology. Respect to the case of pure peer-to-peer architecture, in a super-peer one the number of links among nodes of different stub domains are minimized, and their number does not depend on the total number of nodes but only on the number of clusters into the system. Since in super-peer topologies fewer inter-domain links convey messages exchanged among the nodes, the data dissemination exhibits better quality-of-service in terms of performance and reliability (since data dissemination occurs in the most of the cases using links that belong to the same stub domain where policies to guarantee quality-of-service are applicable), and a wiser usage of network resource (as proved by [124], a super-peer architecture presents a multicast cost tree, *i.e.*, the bandwidth efficiency of the multicast protocol, comparable to the one exhibited by IP Multicast).

3.1.3 Building a Multicast Tree

As stated at the beginning of section 2.2.2, when designing an application-level multicast, one key architectural choice to take is providing a structured organization of the nodes, such as in a tree, or an unstructured one, such as a mesh. Our choice has been to adopt a tree-based solution and to propose mechanisms to overcome its intrinsic weaknesses. The reason of such decision is the following one: as demonstrated in [67], tree-based approaches provide direct control over the path followed by messages and exhibit lower communication overhead. In the approach that we propose in this dissertation, the tree-base application-level multicast is built on top of a structured or *Distributed Hash Table* (DHT) overlay since it simplifies the tree construction. Among the available DHT solutions, we have preferred the ones based on the *Plaxton Mesh* [125] data structure since it enforces a fast search operation (in fact, its complexity is approximately $O(log(N))$ hops, where N is the number of peers in the overlay). Moreover, we have chosen to build our system on top of Pastry [126], which will be described in the following subsection 3.1.3.1, however, other similar DHT overlays can be used since most of Plaxton-based DHTs do not present strong differences among themselves. In addition, we have drawn on the experience of **Scribe** [127] for the tree building process, which is illustrated in the following subsection 3.1.3.2.

3.1.3.1 A DHT overlay: Pastry

Pastry is a peer-to-peer location and routing substrate that forms a robust and self-organizing overlay topology in Internet, where each node is univocally identified by a 128-bit *nodeID*, *i.e.*, a sequence of digits with base $2b$. Pastry realizes the so-called *Distributed*

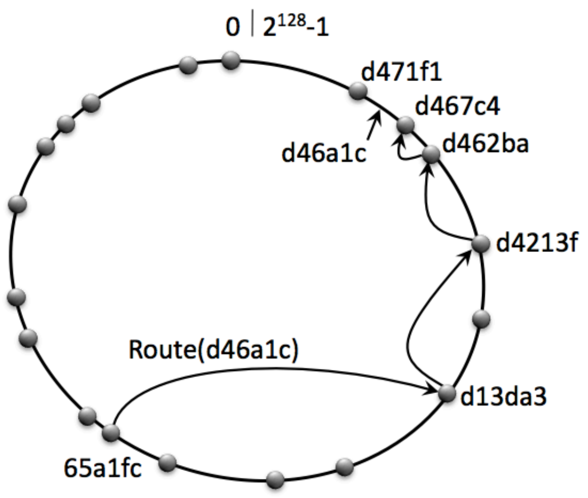


Figure 3.5: Overview of message routing in Pastry

Hash Table (DHT), where nodes are organized in a ring topology ordered by the *nodeID*s, which represent the position of the node in the circular key-space. If a node sends a message, which has attached a key, *i.e.*, a sequence of digits with the same base as the *nodeID*, to the Pastry substrate, the message is routed to the Pastry node with the *nodeID* that is numerically closest to the message key. To perform the routing, each Pastry node stores several list of nodes. The list of leaf nodes of a given node ν contains the $L/2$ nodes with the closest *nodeID*s to the one of ν in each direction around the circle. Then, each node has a routing list that contains the pairs composed of *nodeID*s and network address of the nodes known to the given node. Figure 3.5 illustrates the dissemination of a message in Pastry: at a generic routing step, the following operations are performed:

1. if the key of the message is equal to the *nodeID* of the current node, then the message has reached its destination;

2. given the key of the message, namely $msgKEY$, if the next node on the ring has a

$nodeID$ that shares less digits with $msgKEY$ than the $nodeID$ of the current node, then the current node is the destination of the message (*e.g.*, in Figure 3.5 node identified by $d467c4$ is the destination of message $d46a1c$ since the next node has a $nodeID$ equal to $d471f1$).

3. otherwise, the next node to whom forwarding the message, namely $NEXTnode$, is found based on $SHARE$ that indicates the first digits that the key of the message and the $nodeID$ of the current node have in common:

   (a) the routing list is queried and $NEXTnode$ is the first node that has the $nodeID$ with the same beginning digits of $SHARE$ and has at least one digit in the following positions that matches the value at the corresponding positions of the $msgKEY$ (*e.g.*, in Figure 3.5 if the current node is $d13da3$, the next node would be identified by $d4213f$, which has one more digit in common with $msgKEY$ than the current node).

   (b) if the query to the routing list returned no results, the $NEXTnode$ is found by querying the list of leaf nodes. $NEXTnode$ is the node with a $nodeID$ that begins with the same digits as SHARE, and has the following digit closer than the one in $msgKEY$ (*e.g.*, in Figure 3.5 if the current node is $d462ba$, in the routing tables there are no nodes that have a $nodeID$ starting with $d46a$, so the list of leaf nodes is accessed searching a node with $nodeID$ starting with $d46$ and the 4th digit closer to a. Node $d467c4$ is returned, so the message is forwarded to it, as shown in the figure).

In addition to the list of leaf nodes and the routing table, each Pastry node ν has a neighborhood list which contains the M closest peers to ν, where the proximity is computed in terms of a routing metric, *e.g.*, round trip time. The neighborhood list is used during the routing in order to optimize the forwarding path reducing the delivery time. Pastry is a very efficient solutions, since it is robust to node *churn*, *i.e.*, nodes may suddenly join and/or leave the system. In fact, the entries in the lists of a node are temporary, and keep-alive messages are exchanged among the nodes in order to maintain the relative entries in the lists. However, Pastry does not provide means to detect when a message has been lost by the network and to recover it.

3.1.3.2 A DHT-enabled Multicast Tree

An application-level multicast can use the DHT capabilities of Pastry to construct and maintain a multicast tree, and the widely-adopted solution is the one implemented in Scribe [127], a large-scale and decentralized publish-subscribe infrastructure. Specifically, a multicast session is univocally identified by a *groupID*, *i.e.*, a sequence of digits with the same base as the *nodeID* of a Pastry node. Such *groupID* is used to identify in the system the root of the multicast tree: the Pastry node with the *nodeID* numerically closest to the *groupID* is the root. When a node wants to join a multicast session, it forwards a special message, namely *JOIN_REQ*, that has the *groupID* of the session as key. When a node receives a *JOIN_REQ* and has not joined the multicast session identified by a *groupID* that matches the key of the *JOIN_REQ* (*e.g.*, interaction $1'$ in Figure 3.6), it forwards the message to the next node on the path towards the root. If the *JOIN_REQ* reaches the root, or a

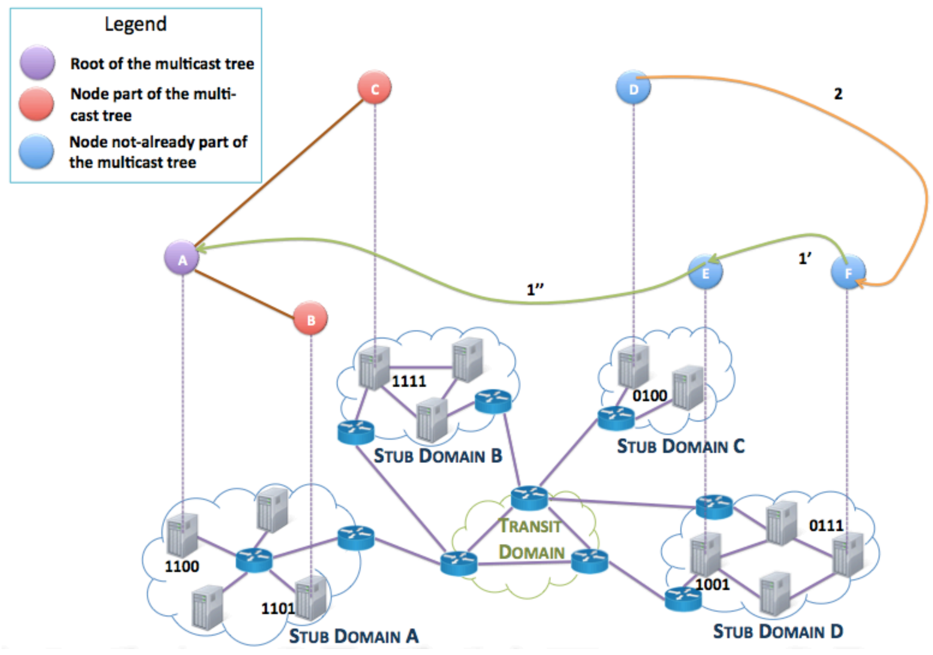


Figure 3.6: Tree building using Pastry

node that has joined the multicast session of interest (*e.g.*, interaction $1''$ in Figure 3.6), the node that received the message stores in its children list the network address of the sender of the $JOIN\_REQ$ and replies with a $JOIN\_ACK$, that contains its network address and the one of the tree root. The node that receives a $JOIN\_ACK$ stores the received network addresses in the parent list. After a node joined a tree, it can send a message in multicast by forwarding it to the tree root (*e.g.*, interaction 2 in Figure 3.6), which will disseminate through the multicast tree. A node can also decide to leave a multicast session, in this case it sends a $LEAVE$ message to its parent, which will cancel the related entry in its children list, and to its children, which will perform the joining procedure. Scribe provides only best-effort delivery, but exhibits functionalities to repair the multicast tree in case of link and node crashes. In fact, nodes periodically send heartbeats to their children, which can detect failed parents if they do not receive the heartbeat within a fixed timeout. In case of a failed parent, its children nodes start the joining procedure by forwarding a new $JOIN\_REQ$ message.

3.1.4 Event Dissemination using the proposed Two-tiers Organization

To sum up our approach for an efficient data dissemination in Internet-scale systems of systems, let us consider the example shown in Figure 3.7, where a node belonging to a given system decides to publish a message of a given topic, and there are several interested nodes belonging to different systems than the one where the publisher resides:

I Operation : Node N_1 publishes the message m_1 that is related to a certain topic, identified with $groupID$ equal to $d46a1c$, so a multicast message is sent to all the interested nodes within the cluster and to its coordinator;

II Operation : After receiving the message m_1, the coordinator C_1 forwards it to the root of the overlay tree identified by the $groupID$ of m_1;

III Operation : The root C_2 exchanges the received message with all the interested nodes in his cluster (arrows $3'$ in the figure) and then it passes m_1 to its children (arrows $3''$ in the figure) in the multicast tree;

IV Operation : All the coordinators reached by the message of the root C_2 perform an IP Multicast in their cluster (arrows $4'$ in the figure) in order to reach the interested nodes and then forward m_1 to their children (arrows $4''$);

V Operation : The message is propagated through the layers of the multicast tree until

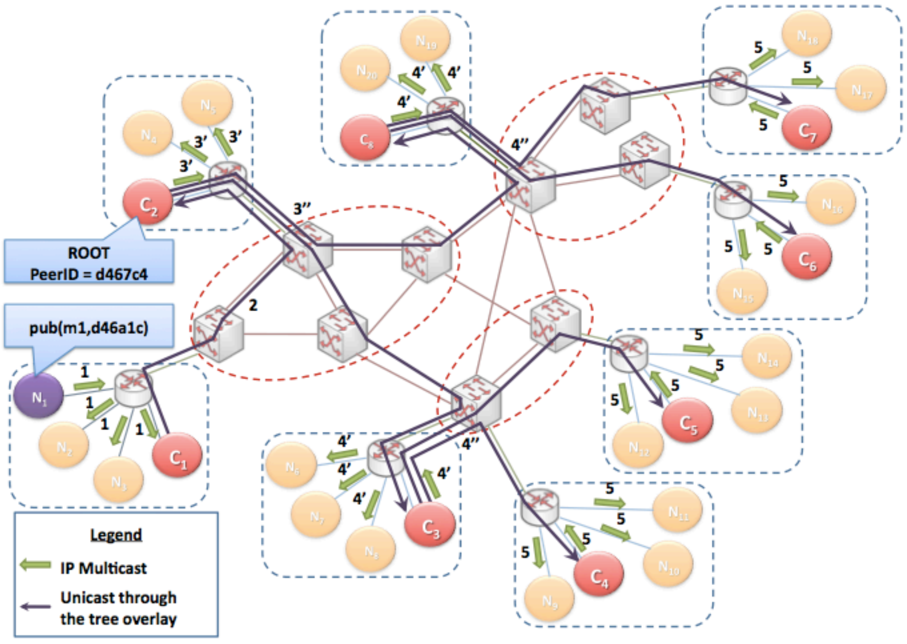


Figure 3.7: Event dissemination through the coordinator tree and the clusters

there are coordinators, such as C_4-C_7, with no children. Subsequently, these coordinators multicast the received message into their cluster, and the delivery process is concluded.

3.2 Reliable and Timely Data Dissemination in TODAI

As presented in the subsection 1.3.3, a publish/subscribe middleware can be affected by loss patterns on the communication links and process/link crashes (as in most of the publish/subscribe services, churn is not directly treated since we consider it as a special case of node crash). We have analyzed where such failures may happen within TODAI and how they affect the correct data distribution, and we have obtained the following considerations, schematically illustrated in Figure 3.8:

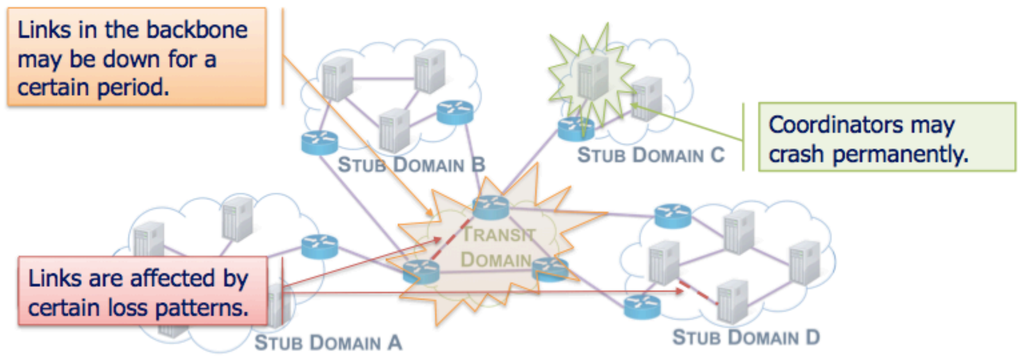

Links in the backbone may be down for a certain period.
Coordinators may crash permanently.
Stub Domain B
Stub Domain C
Transit Domain
Links are affected by certain loss patterns.
Stub Domain A
Stub Domain D

Figure 3.8: Failures occurring in the Two-tiers Organization of an LCCI

Coordinator crashes : The main weakness of our approach is that the entire system is vulnerable when a coordinator fails, however, the crash of one of the other nodes within a cluster does not have any sever consequence. In fact, the failure of a coordinator leads to both the isolation of the affected cluster from the rest of the system and the disconnection of some coordinators from the rest of the overlay tree. Moreover, since the entire traffic toward the outside world passes through the coordinator, the consequent strong workload exacerbates its time to fail.

Message losses : Wide-area networks are characterized by considerable loss patterns [128]. However, not every parts of the TODAI infrastructures are affected by message drops at the same measure. In fact, communications within systems exhibit sporadic losses, caused more by omissions at the application layer than by failures attributable to the network. This is due to the centralized control of stub domains, that allows implementing proper measures to guarantee a reliable communication. The same does not happen for communications among different systems over wide-area networks, where a considerable amount of messages is dropped due to network failures. Moreover, such losses occurred in Internet-enabled data exchanges exhibit a bursty trend, *i.e.*, losses are not isolated but a number of consecutive messages are dropped.

Link crashes in the LCCI backbone : Several studies of the Internet availability, such as [129, 130], 20 percent of all the network failures are not recovered within 10 minutes. In fact, BGP's fault recovery mechanisms usually exhibit a delayed convergence by requiring many minutes before being able to restore the network connectivity after a link or router failure [131]. For the duration of the time needed for the BGP convergence, all the messages disseminated along a link are irremediably lost. We can model such phenomenon by means of transient link crashes. Even if such phenomenon is quite common to occur in Internet, it is unreasonable to happen in stub domains, due to the centralized control that characterizes them. In fact, *Proactive Failure Recovery* (PFR) can be adopted to re-route data traffic to backup paths without waiting for the completion of routing convergence after a local link failure [132].

TODAI supports several strategies to handle such failures without leading to performance degradations. Specifically, in order to avoid disconnections of one cluster to the other ones caused by coordinator crashes, we propose a replication mechanism, which is presented in subsection 3.2.1. With respect to the other failures that can compromise the correct data distribution, we decided to adopt a "divide et impera" approach, since a unique solution does not fit all the parts that compose TODAI. In fact, communications among the nodes within a cluster are realized using IP Multicast, so TLM approaches are more indicated. While communications among different clusters are implemented using an overlay tree-based solution, so ALM reliability means are more suitable. Therefore, we divide the issue of providing reliable communication considering if the messages are exchanged over Internet or over LANs: how to treat message losses in the first case is the focus of subsection 3.2.2, while subsection 3.2.3 talk about the second case.

3.2.1 Coordinator Replication

A well-known solution to tolerate the coordinator crash problem is to replicate the coordinator, however, the choice of which replication flavor to use is critical since it affects the quality of the system perceived by the end-users. On one hand, one option is to use a passive replication scheme [133], in which the failed coordinator is replaced by one of its backups. The case that both the coordinator and all of its replicas fail at the same time is extremely rare, so this solution improves the availability. However, timeliness is compromised since the cluster would be isolated for a certain time window, *i.e.*, the time to detect the failure of the coordinator and election of a new coordinator among the backups, so the overlay tree would be disconnected in some of its parts. On the other hand, an other option is to use an active scheme [134], in which a cluster exposes a virtual coordinator made up of a set of partners with equal responsibilities. Since there are several coordinators available at the same time, a failed coordinator is instantaneously replaced without isolating the cluster or affecting the overlay tree. So timeliness is achieved, however, availability may suffer due to the possibility of failure of all the coordinators due to common mode errors. We propose a **hybrid replication scheme** where the coordinator is actively p-redundant, *i.e.*, there are p coordinators active at the same time, moreover, there are k backups for each active

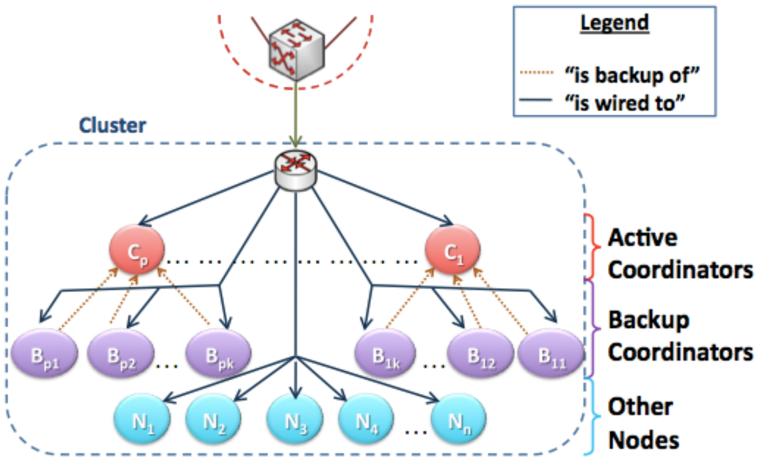


Figure 3.9: Event dissemination through the coordinator tree and the clusters

coordinator. The system designer is free to choose the robustness of the cluster varying $< p, k >$. Replication allows clusters to tolerate coordinator crashes without being disconnected to the outside world for a certain period of time. Moreover, such solution has also a side-effect: a positive impact on the overlay tree management. In fact, since the participants of the tree would result highly available, there is no need to cope with node crashes, avoiding traditional routing approaches that circumvent the failed element and compromise the timeliness of the message delivery.

In literature, several algorithms have been proposed for efficiently electing a unique process to play a particular role in a system, and a there are two common properties guaranteed by all of them [135]: *Safety*, *i.e.*, at the end of the run of the election algorithm only one non-crashed process is elected, and *Liveness*, *i.e.*, all the process participate to the election and get eventually elected. Our application scenario is particularly challenging for an election algorithm since it is characterized by the following features: (*i*) each node has a unique identifier; (*ii*) the nodes that participate to the election fails according to the fail-stop failure model, *i.e.*, a crashed process does not recover; (*iii*) communication among the participants is not considered reliable since there is a not negligible probability to drop a message; (*iv*) there is no certain upper bond on the communication delay; and (*v*) a node may exhibit timing failures, *i.e.*, it responds to incoming messages with an unpredictable delay. Since there are communication failures in a cluster, what an election means needs to be redefined. In fact, a coordinator cannot be unique through the system, violating the safety property of the election. Even if in our approach electing redundant coordinators is exactly the goal to tolerate coordinator crashes, it does not mean that we have to relax the safety property of the election. Not all the active coordinators perform the same actions within the system, as we describe in subsection 3.2.2.2, so we have to elect several active coordinators by performing several election round, where in each one a single active coordinator is elected. Among the classical election algorithms the most feasible one to select a coordinator within a given cluster in TODAI is the **Invitation Algorithm** [136] by Garcia-Molina. Such algorithm is especially attractive for election in case of communication failures since it proceeds by augmentation, which is the opposite of the *Bully Algorithm* that adopt a brute force approach and elects only the node with highest priority to be the unique coordinator. In fact, when dealing with communication failures, Garcia-Molina has proved in [136] that the bully algorithm does not guarantee the safety property.

The Invitation algorithm is based on the concept of group, *i.e.*, a set of nodes that elects the same coordinator, and on a continuous merge procedure, where coordinators periodically try to combine their group with other ones in order to form larger groups. At the beginning, each node belongs to a singleton group where it plays the role of coordinator. Periodically coordinators perform an announce procedure, *i.e.*, they broadcast a keep-alive message to the others nodes for two reasons: (*i*) to announce that it is correctly running, and (*ii*) to invite others nodes to join its group by executing a merge procedure. If the receiver has a lower id, it joins the group and considers the sender its coordinator. Otherwise, it notifies that it has a greater id, and forces the other node to give up the role of coordinator and to consider itself the new coordinator. In order to build the organization illustrated in Figure 3.9, we perform several election rounds, each one has assigned a unique IP Multicast address to exchange election and keep-alive messages. To avoid that the same process can be elected coordinator in distinct election rounds, we impose that a process with higher id force other nodes to give up only in one round, while in the others it spontaneously assume the other process, characterized by a lower id, as its coordinator. In addition, each process is characterized by a vector of process id that is filled during the merge procedure: when a process receives a keep-alive message from another one with a greater id, it do not only give up being a coordinator, but its id is inserted in the vector of the new coordinator if there is still space.

In case an elected coordinator fails and its backups do not receive on time the keep-alive message, a recovery procedure is activated. Specifically, the backup with the greater id is elected new coordinator (such operation is really fast since a backup knows the ids of the other ones and can quickly decide if it is able to be the next coordinator). The process that is identified by a randomly chosen id of the ones contained by the vector of the new coordinator and not already acting as a backup is selected as backup to replace the backup that became the new coordinator.

3.2.2 Reliable and Timely Inter-Cluster Communication

Communications among nodes in different clusters are conveyed by Internet, so they are affected by link crashes and bursty loss patterns. In this section, we propose proactive techniques able to provide reliable and timely data dissemination by tolerating such failures. Specifically, in the subsection 2.3.2, we have evaluated the efficiency of the different techniques available in the current literature and shown that only two proactive techniques are able to provide suitable trade-off of reliability, timeliness and scalability: distributed FEC and multiple tree disseminations. In subsection 3.2.2.1, we propose an innovative protocol that embodies distributed FEC to tolerate losses, while in subsection 3.2.2.2 we illustrate a novel strategy that is aware of the underlay topology when building multiple diverse trees and disseminate information using the built multiple trees. However, none of these techniques can tolerate jointly link crashes and losses, as shown in Table 2.1. Therefore, we present in subsection 3.2.2.3 an integrated approach to combine distributed FEC and multiple tree distribution to achieve a reliable and timely event dissemination despite of the occurrence of link crashes and message drops. Last, we observed in Figure 2.15 that proactive techniques are not able to achieve an high degree of reliability as the reactive ones do. To increase the reliability exhibited by the dissemination infrastructure we have combined the proactive methods presented in this section to a gossiping algorithm. This matter is described in details in subsection 3.2.2.4, which conclude this section.

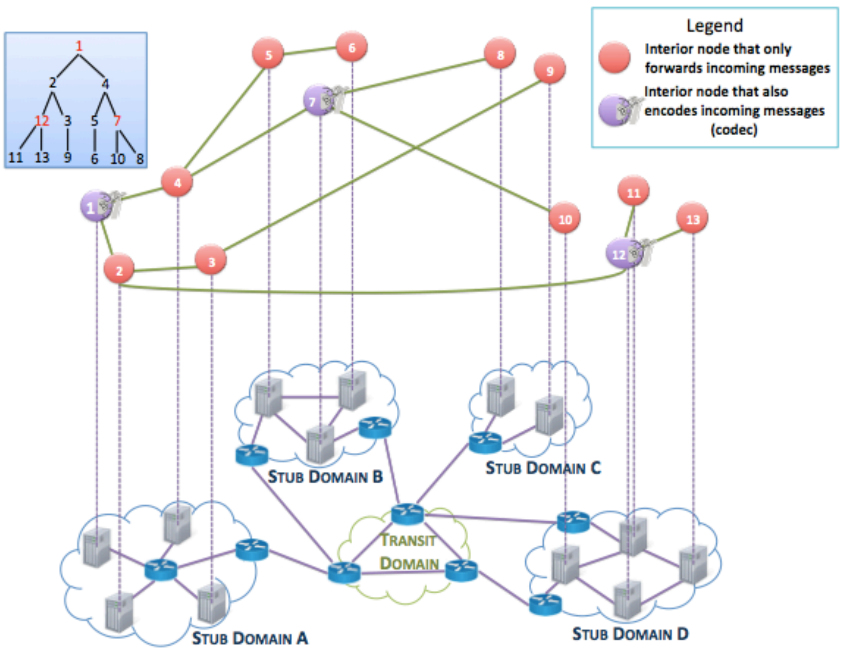

Legend
Interior node that only forwards incoming messages
Interior node that also encodes incoming messages (codec)
Stub Domain A
Stub Domain B
Stub Domain C
Stub Domain D
Transit Domain

Figure 3.10: Overview of the Selective Network Embedded FEC

3.2.2.1 A Distributed FEC Approach to Tolerate Losses

Selective Network Embedded FEC consists of enabling a subset of interior nodes, called *codecs* and illustrated in Figure 3.10, to perform encoding operations in order to adapt the applied redundancy degree to better face the loss patterns exhibited by subtrees delimitated between two codecs without implying any fluctiation in the data delivery time. A method to select proper codecs within a multicast tree in order to maximize reliability and minimize performance overhead has been formulated for the first time in the approach called *Network Embedded FEC* (NE-FEC) and presented in [137]. Since the issue of deciding where to locate encoding belongs to a class of optimization problems known as the *P-Median*

Problem[3] [139], which is $NP - Hard$ on general networks for an arbitrary p (where p is a variable) and polynomial when the network is a tree, NE-FEC uses a greedy algorithm to quick resolve the problem. Specifically, the algorithm is executed by assuming that the number of needed codecs, namely p, is somehow known beforehand and its run is structured in the following three stages:

1. each interior node periodically estimates the loss rate experienced by outgoing channel towards one of its children, and these estimations are regularly collected in a centralized place (*i.e.*, the root of the forwarding tree);
2. after receiving the loss estimations from all the nodes, the root performs the greedy algorithm on the collected data and the codecs are placed within the multicast tree by iteratively resolving reduced, simpler, sub-problems (*i.e.*, the best location for the first codec is determined, then the location for the second codec and so on, until all the codec are placed);
3. the selected nodes are informed to perform FEC operations by applying proper redundancy degree, which is also quantified by the root on the basis of the received loss estimations.

In literature there is available another approach, described in [140], named *Multi-hop FEC* (MO-FEC) and characterized by several similarities to NE-FEC (*e.g.*, it is also based on a

[3]Specifically, Selective Network Embedded FEC is a case of the discrete version P-Median problem introduced by Hakimi in [138]. The absolute P-Median problem consists of finding medians among existing points in order to optimize the objective function. When P-Median is applied to a graph, medians can lie anywhere along the graph's edges, but Hakimi demonstrated that there is always a collection of p vertex of the graph that optimize the objective function, although it may not be the optimal. So it is introduced a discrete version where the p medians are looked for only among the vertex.

greedy algorithm to decide where FEC needs to be performed). These approaches exhibit an evident drawback that strongly limits their practical applicability to the case of Internet-scale data dissemination infrastructures: all the optimization operations are performed by the root in a centralized manner, only after building the knowledge about the system topology and the loss patterns affecting its links. Although centralization of the decision making process has the strength to simplify the solution and guarantee to take optimal decisions, such strategy is unfeasible since collecting global information in a large-scale distributed system is extremely difficult, if not impossible. In addition, even if we assume possible acquiring global knowledge of the system, they still exhibit serious scalability limits that prejudice their usage to systems composed by a large number of nodes. In fact, the time to collect loss statistics in a centralized point of the overall system linearly increases with the number of nodes composing the multicast tree. Therefore, the memory required to store all the collected data and/or the load to resolve the optimization may overwhelm the resources of the root, so that it becomes unable to take any decision at all. On the other hand, the choice of using a greedy algorithm is not opportune since such algorithms mostly fail to find the globally optimal solution, because they usually do not operate exhaustively on all the data [141].

A distributed approach has the strength to overcome the applicability issues for large-scale systems that afflicted the previous solutions since codec placement is performed using local computations at each node of the multicast tree avoiding turning to a central decision maker with global knowledge of the system. However, this advantage is paid at the expenses of obtaining a placement that is slightly far away from the optimal one in terms of achievable performance and resiliency. Neglecting such drawback, both the previous approaches have been extended by introducing a distributed procedure for the placement of the codecs. In the distributed NE-FEC [142], each node of the multicast tree estimates the redundancy degree needed to successfully receive a message from its parent and the number of its children that are unable to decode the received messages. Periodically it delivers such information to its parent in order to perform the placement of the codecs and their adaptation to the changing conditions of the network. On the other hand, the distributed version of the MO-FEC approach [143] differs from the previous one: (*i*) it takes a proxy-based prospective rather than a P2P one (*i.e.*, the interior nodes are not the end users of the incoming messages, but they are special-purpose data service nodes with certain functions, such as coding, monitoring and adaptation), (*ii*) coding is places at the extreme of congested links (*i.e.*, those links that do not have available bandwidth). However, such approach provides less guarantees to reduce the total number of codecs than the previous one. Therefore, we preferred to use the solution of [142] as basis for our approach. Specifically, when disseminating a message between two nodes within the multicast tree, the destination has to forwards a NACK message to notify the source that the message has not been correctly delivered[4] and to indicate the number of lost packets. Each node in the tree is characterized by a variable named $UNREACHED\_KIDS$, which indicates the number of its children that could not receive a message and another variable called $LOSS\_LENGTH$, which contains the maximum of the lost packets communicated by each of its children. When an

[4]A message cannot be delivered if the number of lost messages is greater than the correcting capacity of the adopted FEC techniques. In case all the packets are lost, such event can be detected by the destination only when he start receiving packets belonging to another message with an identifier greater than the expected on. Even if the loss detection is not performed on time, the effectiveness of the algorithm is not affected.

interior nodes that has been able to decode the received message, is not acting as a codec and has the $UNREACHED\_KIDS$ not null, it represents a candidate to be a codec. The condition that a candidate has to satisfy in order to be chosen as a codec is to have $UNREACHED\_KIDS$ greater than a given threshold σ. If such condifion is not met, it can not be a codec and notifies the codec that encoded the received message its value of $LOSS\_LENGTH$, asking to increase the applied redundancy. In case the node has been elected as codec, it starts applying to the incoming messages an additional redundancy equal to $LOSS\_LENGTH$, or can even ask the previous codec to reduce its redundancy.

To provide a better comprehension of this algorithm, let consider the example illustrated in Figure 3.11, where messages need to be disseminated in a tree of fifteen nodes. At the dissemination of the first message, namely m_1, the only node in charge of applying FEC is the root of the tree, and the adopted redundancy degree, indicated as ρ_1 in the figure, has been opportunely chosen according to measurements of the path loss along the links towards its children[5]. Let consider that the root needs to disseminate a message, refer to the upper part of the figure. All the nodes will experience a certain loss pattern, *i.e.*, total number of packets lost during the dissemination of m_1, but some of them have successfully decoded the message while other ones could not. The latter ones notify their parents of such losses, so that some nodes has the variable $UNREACHED\_KIDS$ that is not null. Let consider the case of the threshold σ fixed at the value of 3, in figure there are nodes that cannot be

[5]The redundancy degree is chosen greater to the maximum loss pattern $\rho = max_i(p_i) + \epsilon$, where p_i is the loss pattern on the link towards the i-th children, while ϵ is a properly-chosen positive integer. The redundancy is not equal to the maximum loss pattern since loss patterns are highly variable so we apply over provisioning in order to tolerate slight variations in the loss pattern. The integer ϵ is randomly chosen within $[1\ ,\ max_i(p_i)\ /\ 2]$ and represents how many losses can be tolerated over than the expected ones.

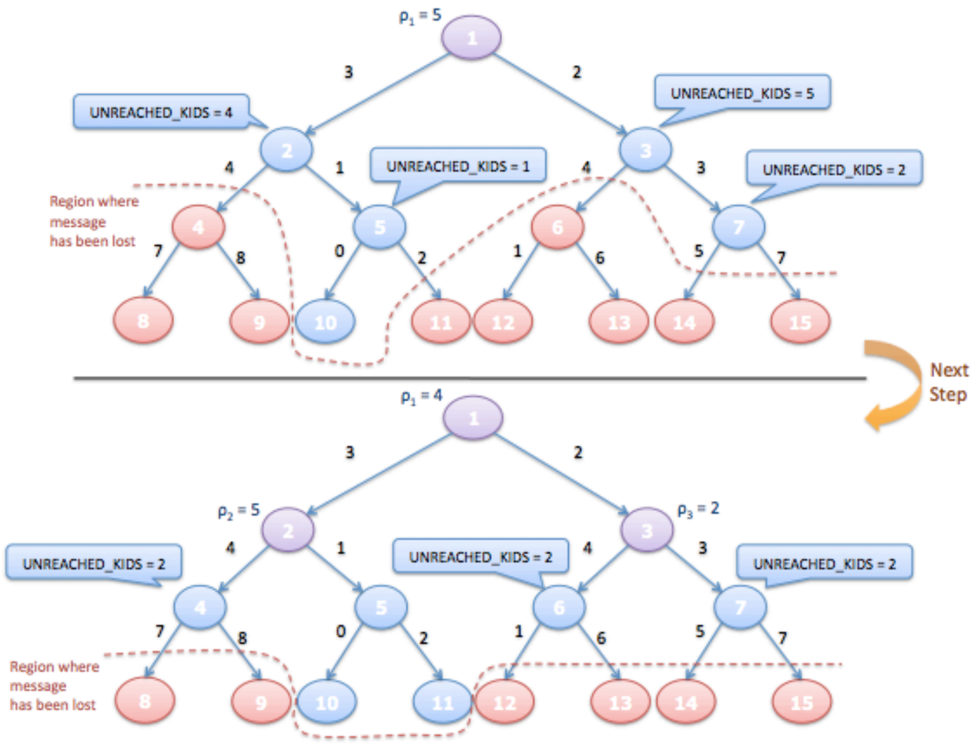


Figure 3.11: Application of the algorithm to elect codes in a multicast tree

codecs since the value of their $UNREACHED\_KIDS$ is lower than σ. In this case, they notify their parents that some of their grandchildren have lost the message m_1, therefore, the value of $UNREACHED\_KIDS$ of such nodes is incremented by taking into account the number of grandchildren that have experienced the message loss (*e.g*, this is the case of nodes 2 and 3: the number of their children that have lost the message is respectively equal to 1 and 2, but the value of $UNREACHED\_KIDS$ is respectively set to 4 and 5 since also grandchildren are considered). On the other hand, there may be nodes whose value of $UNREACHED\_KIDS$ exceeds σ, and for this reason they are elected codecs. In the dissemination of the next message, namely m_2, the tree presents three codecs, as shown in the lower part of Figure 3.11, each one applying a proper redundancy to the incoming messages. Respect to the dissemination of the previous message m_1, the nodes that failed to receive the message m_2 has decreased. Since the region within which the message has been lost still contains nodes, there are some nodes with $UNREACHED\_KIDS$ not null, however, as shown in the lower part of the Figure 3.11, their value is not greater than σ. Therefore, the number of codecs within the system cannot be further increased, so to guarantee the delivery of the next message, namely m_3, the existing codecs has the only choice to increase their redundancy by taking into account the losses experienced also by their grandchildren.

An important aspect that influences the quality of a FEC technique is the coding technique adopted to generate the redundant data from the information data. Without going into details on the different coding techniques proposed in the last years, since this topic is addressed in Appendix A, here it is sufficient to say that each technique is characterized by a certain encoding and decoding overhead, *i.e.*, respectively the time to generate redundant data and to reconstruct the original data from the received one, and correcting capability, *i.e.*, how many losses can be tolerated. In the years mostly all the proposed FEC protocols, included [142], focused on a certain coding method, called Reed-Solomon code [60], that presents the highest capability and an acceptable overhead for messages of small size. In the case of a distributed FEC algorithm, this is not a winning choice: such code does not support a progressive encoding. Let consider the dissemination in the lower part of Figure 3.11, when using Reed-Solomon code, a codec, such as node 2, needs to perform the following operations in order to increase the redundancy applied to the packet stream: (*i*) it has to acquire the original message (*i.e.*, receive a sufficient number of packets and decode it), and (*ii*) it can apply the increased redundancy by re-encoding the message with a redundancy equal to the sum of the previous applied redundancy and its own one (*e.g.*, in figure node 2 apply a redundancy equal to 9). This causes an incredible drop in performance, making such codes unsuitable for distributed FEC that ams to provide timeliness jointly to reliability. In the recent years a new class of codes, defined *rateless*, allows a progressive encoding of a message. Specifically, adopting a rateless code, a codec as node 2 can forward to its children the incoming messages as soon as they arrive and re-encode the received encoded data to generate new encoded data (*e.g.*, node 2 will encode with a redundancy degree equal to 5 rather than 9 as in the case of Reed-Solomon code) without going back to the original message. Only rateless codes are suitable for distributed FEC when timeliness is a matter, but their usage is protected by patents. Thanks to a collaboration with Digital Fountain[6], which holds the patents for all the rateless coded proposed so far, we have been able to use such codes in our approach.

3.2.2.2 Data Dissemination through Multiple Trees

In the recent years several studies, such as the Skitter Project conducted by Caida[7] in 2006, have been conducted with the scope of analyzing the topological characteristics of Internet. In the context of such studies, one of the objectives was to assess the redundancy degree of paths among two nodes in Internet. On one hand, [144] has demonstrated that several redundant paths exist among the nodes inside a stub domains. On the other hand, [145] proved that nodes on different stub domains are interconnected by more than one path.

[6]www.digitalfountain.com/
[7]www.caida.org/tools/measurement/skitter

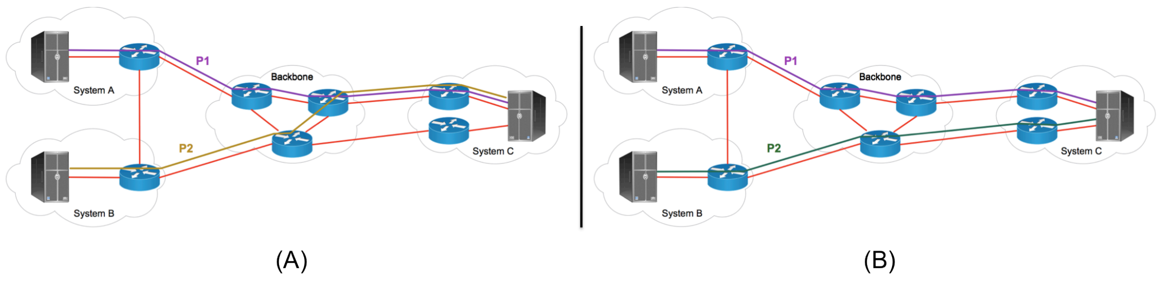


Figure 3.12: Two distinct paths among two nodes placed in two different systems: (A) the paths share a link and (B) the paths are diverse

Therefore, it is possible to conclude that there is strong potential to achieve an high redundancy degree for data dissemination among a super-peer organization such as TODAI [102]. So, as presented in subsection 2.2.2.6, an optimal approach for tolerating failures and guaranteeing a reliable event dissemination is to take advantage of such redundancy. There are three alternative approaches to implement such a solution [112], among which we have chosen a **multiple-tree approach** since it is able to reduce delivery ratio and to better cope with stringent real-time deadlines [116].

As discussed at the end of subsection 2.2.2.6, strategies based on path redundancy enforce reliability and timeliness under the condition to assure diversity among the several paths that interconnect a source to a given destination. When a message has to reach a node from another one, it passes through a succession of network devices, *e.g.*, routers and/or switches. Specifically, given two paths P_1 and P_2 that interconnect a pair of nodes, the following two different situations can present: (i) the two paths share some network devices (as in the left side of the Figure 3.12), or (ii) the two paths do not share any device (as in the right side of the figure). So, we can define a measure of their reciprocal diversity, namely $Q(P_1, P_2)$, as the number of overlapping network devices. Therefore, when paths share any devices, such measure will have a positive value (with respect to the case illustrated in the left side of the figure, we have $Q(P_1, P_2) = 2$), while the two paths are deemed to be diverse if $Q(P_1, P_2)$ is zero (as in the case illustrated in the right side of the figure). The overlapping devices can be identified through measurements at path- and AS- level as described in [146].

How path diversity is assured is a crucial aspect to consider when designing any dissemination strategy based on path redundancy. Traditionally, path diversity has been used as a synonymous of *edge disjoint paths*, *i.e.*, considering a graph G, diverse paths are the ones that do not share any edge. Such definition has been matured within the research community on networking [147], where a networking infrastructure is modeled as a graph with nodes and edges respectively representing networking devices and the interconnections among them. Such approach to the path diversity has also been used in the context of overlay networks [148]. However, building overlay paths that do not share any overlay link does not guarantee the path diversity expressed as the measure Q of their reciprocal diversity equal to zero. In fact, since in most of the cases the overlay network is not aware of the underlay network, paths that do not share elements at the overlay tier can share network devices at the networking layer. Therefore, in TODAI we do not adopt only the strategy of link disjoint paths when building multiple trees, but we propose a novel strategy to define *topology-aware disjoint paths*, *i.e.*, overlay paths that are disjoint both at the overlay and underlay layer. Specifically, given the diversity measure that we have previously introduced, it is possible to formulate the path diversity as follows. Given a forest of n overlay trees, namely T_i with $i = 1...n$, such a forest verifies the path diversity constraint if and only if it is not possible to find in any of the n trees two paths that exhibit a positive value as measure of their reciprocal diversity (*Global Diversity*):

$$\begin{aligned} F = \bigcup_{i=1,\ldots,n} (T_i) : F\ is\ diverse \Leftrightarrow \\ \nexists P_i, P_j \in F : (Q(P_i, P_j) > 0) \wedge (i \neq j). \end{aligned} \tag{3.1}$$

If we want to use such definition to build a forest of diverse trees, the only viable approach is to collect in a central location of the system all the topology information about the system, to formulate the issue of selecting links in each tree with a minimal reciprocal diversity as an optimization problem, which has been proved to be $NP - Hard$ [149], and to resolve it using heuristics. In large scale networks, such centralized approach is not suitable since they exhibit sever scalability limits, as said in the previous subsection when we talked about the weakness of a centralized approach to implement Selective Network Embedded FEC. In addition, it is also impossible to verify if trees satisfy this condition since it requires global knowledge of all the connections among the coordinators in the systems and their reciprocal diversity. Hence, to use a multiple-tree approach in a distributed manner, we have to provide a different formulation of diversity: given a node N_A and n trees, namely T_i with $i = 1, ..., n$, the forest of n trees verifies the path diversity constraint if and only if all the paths from N_A to its parents and children in the i-th tree, namely $P_{i \mid N_A}$ do not exhibit a positive value as measure of their reciprocal diversity (*Local Diversity*):

$$\begin{aligned} F = \bigcup_{i=1,\ldots,n} (T_i) : F\ is\ diverse \Leftrightarrow \\ \forall N_A \ \nexists P_{i \mid N_A}, P_{j \mid N_A} : (Q(P_{i \mid N_A}, P_{j \mid N_A}) > 0) \wedge (i \neq j). \end{aligned} \tag{3.2}$$

where $P_{x \mid y}$ is a path from node y to node x. The local diversity does not imply the global diversity, however, a reduced diversity is the price to pay in order to make possible implementing a distributed algorithm able to construct diverse multiple trees.

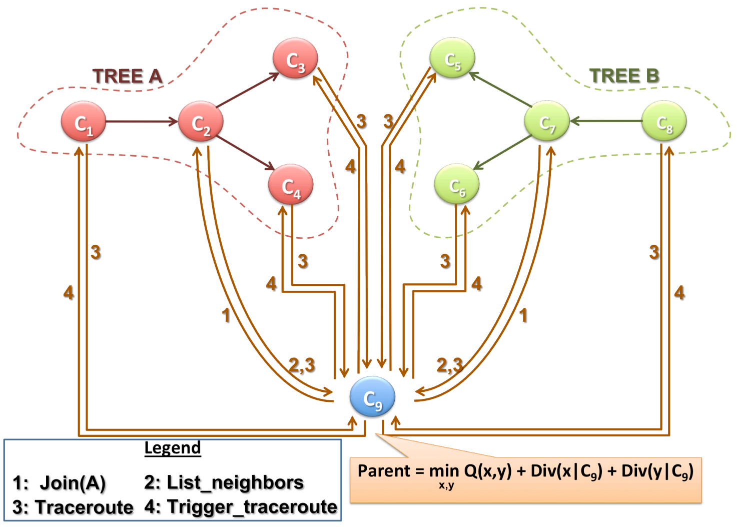


Figure 3.13: Innovative joining procedure to build diverse multiple trees

We have modified the joining procedure described in subsection 3.1.3.2 in order to construct multiple path-disjoint trees that satisfy the local diversity constraint expressed in Equation 3.2. Let consider the situation illustrated in Figure 3.13 where a new node C_9 wants to join two different trees, namely A and B. At the beginning, C_9 sends two join messages through Pastry and two nodes of the systems, indicated in Figure as C_2 and C_7 are contacted. C_2 and C_7 have respectively joined tree A and B. Each one of these nodes replies C_9 with a message containing (i) a list of the neighbors in the tree (*e.g.*, its parent and children) and details on their path, and (ii) content of a traceroute on the path to C_9, *e.g.*, the list of the traversed network devices. Then, C_9 contacts all the nodes in each on of the received lists, and receives traceroute messages about the path to them, too. After collecting such information, C_9 can make the decision on which will be its parent in each tree, according to these two rules:

- the paths from C_9 to its parents have to expose the lowest measure of diversity;

- given a parent of C_9, the paths to its children have to maintain a measure of diversity closer to the value they had before the inclusion of C_9 as a child.

So the node C_9 decides on its parents by performing the following optimization to achieve local diversity:

$$\widehat{x} = \min_{\bar{x}, y=C_9} \left[\check{Q}(\bar{x}, y) + \sum_{x_i \in \bar{x}} Div(x_i|y) \right], \tag{3.3}$$

where $\bar{x} = \{x_1, x_2, ..., x_n\}$ is a list of the possible parents for C_9, while $\widehat{x}$ is the list of the chosen parents for C_9. Moreover, the first addend of the sum to be minimized formalizes the first rule, where $\check{Q}(\bar{x}, y)$ measures the diversity of the paths from node y to the parents contained in vector $\bar{x}$, whose length is equal to n:

$$\check{Q}(\bar{x}, y) = \sum_{\substack{x_i, x_j \in \bar{x} \\ i \neq j}} Q(P(x_i, y), P(x_j, y)), \tag{3.4}$$

where the path from a node x to a node y is indicated as $P(x, y)$. While, the second addend formalizes the second rule and evaluates the variation of the diversity of the neighbors of a node x if y is promoted as child of x:

$$Div(x|y) = \check{Q}(\{V_x \cup y\}, x) - \check{Q}(V_x, x), \tag{3.5}$$

where V_x is the list of the neighbors of node x before putting y in the children list. In the optimal case, due to the locality of Pastry, we will always find nodes that exhibit a diversity measure equal to zero, however, in the real case this is not always possible. Since, the achievable diversity is always lower than the intrinsic diversity of the topology at the network level, we acknowledge that there are cases where the minimum of the previous optimization is not zero.

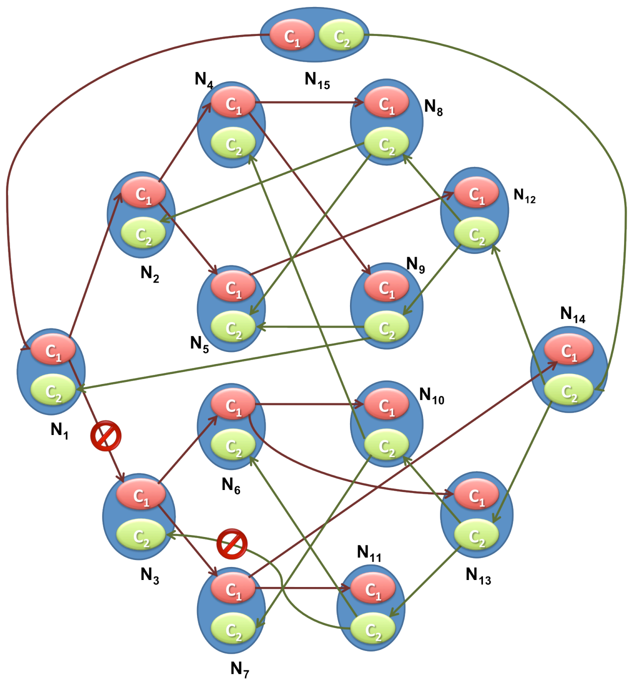


Figure 3.14: Event delivery among coordinator through multiple trees

Since clusters host multiple active coordinators, there is a problem on how performing the joining procedure. In order to avoid the case of coordinators of a same cluster exhibiting different dependencies in the same overlay tree, only one coordinator at a time performs the joining procedure. Recalling from the previous example, at the end of the join procedure, C_9 perform the following operations: (i) sends to its parents the IP addresses of its partners and backups, (ii) receives from the parents the IP addresses of their partners and backups, and (iii) forwards the received IP addresses to its partners and backups. So, each active coordinator will be in charge of sending messages only through one tree. Thus the parameter p also defines the number of multiple trees that are established in the system. Even when using a multiple-tree approach, there may be cases in which some nodes may experience message losses due to link crashes. As shown in Figure 3.14, since two links have crashed, node N_3 will never receive messages published by N_{15} even if it is not isolated. This is possible when all the inbound connections to the link have crashed. However, the outbound connections are still correct[8], and they can be used to recover from this situation.

Messages exchanged though different trees do not reach a node at the same time. Let us consider node N_7 in Figure 3.14. It would receive a message before from N_3 and then from N_{10}. So, when N_7 receives a message from N_{10}, but nothing from N_3 before a timer has expired, it can assume that the message has not reached N_3 and notifies it that it has received a message. So, N_3 knows that it has missed a message and asks N_7 for its transmission. Since it has lost a message, N_3 assumes that all his inbound connections are incorrect and executes the joining procedure to restore its connections to the trees.

3.2.2.3 Integration of Multiple Trees and Distributed FEC

Each of the two proactive techniques illustrated in the previous subsections does not cover all the failures that may occur during the delivery of a message over Internet, but is tailored on a specific class of faults:

1. Selective Network Embedded FEC is particularly effective for tolerating loss patterns that characterize the edges of a multicast tree, but is vulnerable to link crashes. In fact, if an edge crashes and some nodes result disconnected from the rest of the tree, FEC cannot guarantee the message delivery for the nodes in the disconnected part.

[8]We do not treat the case in which all the connections to a node, inbound and outbound, are crashed, because in this case the node would be completely isolated. Since we have assumed that the network is not partitionable, this case can never happen.

2. Multiple-tree represents a powerful mean to tolerate link crashes since it allows circumventing the crashed link and making all the nodes reachable even if link crashes occur. However, multiple-tree is not so effective when the dissemination infrastructure is affected by losses. In fact, it is not negligible that a given packet of the stream may be dropped by all the paths from the producer to a destination. To lower such probability, the message publisher can adopt several strategies that imply a different redundancy degree to the packet flow exchanged between source and destination:

    (a) the most effective strategy is to send replicas of the message through all the trees, but this presents a troublesome side-effect: it generates a strong traffic load that can strengthen the loss patterns experienced by the network;

    (b) a more network-friendly solution is to generate redundant packets by using FEC technique, and to forward a portion of the encoded packets per each tree.

    Even if a FEC-enhanced Multiple-tree approach can theoretically reduce the probability that a node experiences the loss of a given packet, in practice it is not achieved since opportune tuning of the redundancy degree applied by the message producer is not possible. In fact, an optimal tuning of FEC requires gathering global knowledge of the loss patterns along all the links within the system, and this is impractical for Internet-scale systems.

Since they present a dual behaviour, *i.e.*, one technique is vulnerable towards failures that are effectively tolerated by the other one and viceversa, a suitable solution to have a dissemination strategy that proactively tolerates both link crashes and message losses is combining them. Specifically, in each tree the Selective Network Embedded FEC is performed so that the packets dispatched by the root along the tree has an high probability to be delivered to all the nodes despite of losses under the assumption that no link may crash. On the other hand, the producer of a message applies a coding technique to generate r redundant packets from the k information packets (so that the packets to deliver to each node is $n = k + r$). Then, it equally disseminates the n packets though the t multiple trees (*i.e.*, each tree conveys a number of packets equal to n/t). If there may happen only a single link crash that compromises the message delivery along a single tree, each node will receive only $n - \frac{n}{t}$ packets, so the system will tolerate the crash if the number of received messages is greater or equal to the capacity, namely C of the adopted coding technique:

$$n - \frac{n}{t} \geq C \Rightarrow \left(1 - \frac{1}{t}\right) \cdot n \geq C \Rightarrow \left(1 - \frac{1}{t}\right) \cdot (k + r) \geq C \tag{3.6}$$

Considering Equation 3.6, we can formulate a condition on the applied redundancy degree r so that a single link crash can be tolerated:

$$r \geq \frac{t}{t-1} \cdot C - k \tag{3.7}$$

This result can be generalized for the number, namely ft, of faulty trees, *i.e.*, they do not deliver their packets to a certain node due to a link crash, as follows[9]:

$$r \geq \frac{t}{t-ft} \cdot C - k \qquad iff \quad ft < t - 1 \tag{3.8}$$

Equation 3.7 shows that the tuning of the FEC coding used at the information producer does not depend on the loss patterns affecting the network, but on the tolerance degree that

[9]When ft is equal to t, *i.e.*, we aim at tolerating the case where all of the multiple trees, expect one, can be faulty without leading to a message loss, the only viable strategy is to send a copy of the message per tree.

the system has to exhibit. This means that the tuning does not require a global knowledge of the system so it can be realized in an Internet-scale system.

3.2.2.4 A reactive technique teamed up with proactive techniques

As shown in Figure 2.15, proactive techniques do not make possible to achieve the same level of reliability that reactive methods are able to provide. To further increase the reliability level achievable by TODAI, we propose to team up the proactive techniques illustrated in the previous sections with a reactive gossip-based recovery mechanism. The goal of using gossiping is to allows nodes to detect lost messages and rapidly recovery them. Specifically, at a random time, each node decides to perform a gossiping round to a set of randomly chosen neighbors. After deciding the communication partners, an important aspect in designing a gossip- based protocol is defining the content of the message that each node sends to the communication partners during a gossip round. As discussed in subsection 2.2.2.4, in the current literature there are three main approaches: a gossiper decide to send (*i*) all the received messages since the end of the previous gossip round (*Push approach*), or (*ii*) the identifier of the least messages so that the receiver can trigger a retransmission in case of losses (*Push/Pull approach*), or (*iii*) a request for information on recent received messages to use for loss detection and retransmission triggering (*Pull approach*). Gossip strategies that require retransmissions are not suitable for our case, since the time to recover lost messages is strongly affected by the network dynamics, so the only viable strategy is to use a push approach. However, pushing all the recently-received messages generate a considerable traffic load that can generate congestion and overloading phenomena within the system. To still use a push approach without incurring in any network unfriendly behaviors, we adopted a novel strategy based on the idea of *Random Linear Coding* (RLC) and called *Algebraic Gossip* [150]. Specifically, messages are viewed as vectors in a Galois Field[10] $GF(2^L)$ with size L and the content of a gossip message, namely M_{gossip} is generate as linear combination of k least recent messages with k randomly-chosen elements of $GF(2^L)$:

$$M_{gossip} = \sum_{i=1}^{k}(a_i \cdot m_i) \qquad a_i, m_i \in GF(2^L) \tag{3.9}$$

At each gossiping round, the gossipers spread messages containing the result of Equation 3.9 and the list of coefficient a_i used to perform the linear combination. Each node starts receiving such messages from which they detect and recover losses. In fact, if a node has lost a message, *e.g.*, message m_k, it can recover it with only one gossip message by resolving:

$$m_k = \frac{M_{gossip} - \sum_{i=1}^{k-1}(a_i \cdot m_i)}{a_k} \tag{3.10}$$

where m_i with $i = 0, ..., k-1$ are the messages that the node has already successfully received. General speaking, if the node has lost several messages, *e.g.*, l messages failed to reach the node, they can be recovered when l gossip messages are delivered to the node by resolving the following system of linear equations:

$$M = A^{-1} \cdot G \tag{3.11}$$

where A is the matrix that contains the coefficients a_i, M is the matrix of the messages to recover, while G is the matrix of the received gossip messages. Using RLC, [150] has proved that Algebraic Gossip allows to recover mostly of the dropped messages within a single gossip round without requiring any additional retransmission.

[10]Refer to Appendix A for more details of Galois Fields and linear coding.

3.2.3 Reliable and Timely Intra-Cluster Communication

Communications among nodes within the same cluster are realized using IP Multicast, so, as stated at the beginning of this section, they are affected by loss patterns. As shown in Figure 2.14, retransmission-based approaches are able to guaranteeing strong reliability assurances at the expenses of timeliness, while FEC-based ones exhibit an opposite characteristic. Combined solutions are the only one that allows a good trade-off among reliability and timeliness, for this reason the approach that we proposed for a reliable inter-cluster communication jointly uses retransmissions and FEC. We illustrate in subsection 3.2.3.1 a novel FEC technique that aims at dynamically adapting the applied redundancy to the needs of each destination without adopting a "one measure fits all" perspective. While, in subsection 3.2.3.2, we describe how calibrating the redundancy degree applied by the multicaster and increasing the reliability of the previous FEC technique by taking advantage of neighboring nodes.

3.2.3.1 Layered FEC

In a classic FEC technique, as the one described in subsection 2.2.1.4, the duty of coding an outgoing message is assigned in a centralized fashion to the message multicaster. The multicaster has to face a critical choice for the effectiveness of the communication: how many redundant packets are needed to ensure delivery to all destinations? Typically, the redundancy degree is set equal to the maximum loss burst experienced on the path between the multicaster and one of the destinations, but this is not a scalable choice due to the "the boat unbalanced by the heaviest" problem. In literature, this matter has been addressed

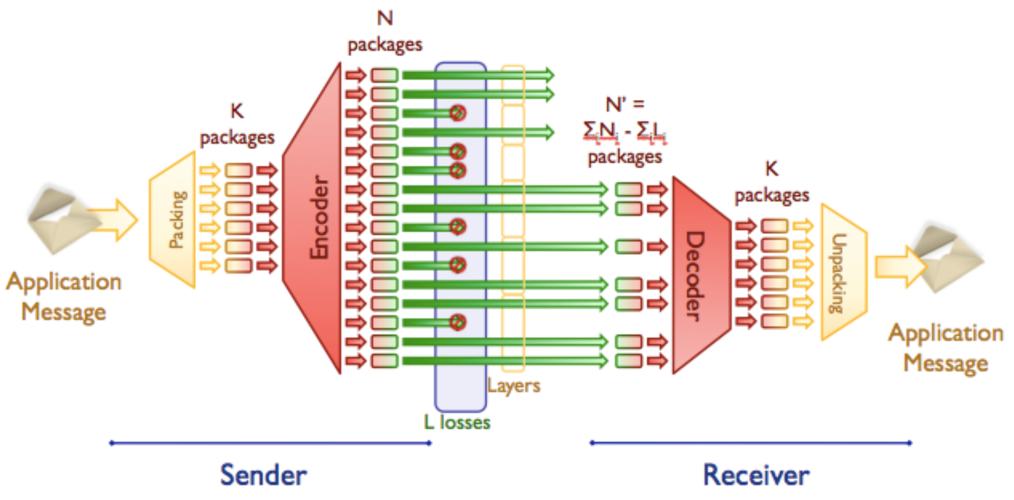


Figure 3.15: Overview of a layered FEC

only by placing the FEC encoding on the receivers, as in the Ricochet protocol [81], but this reduces the timeliness offered by the protocol. To use FEC at the multicaster without incurring in any scalability issue, we propose to use a **Layered organization of the FEC** data [151], shown in Figure 3.15: non all the packets generated by the encoder are distributed using a single multicast session, but packets are grouped in layers, each one representing a multicast session. A destination has to subscribe to as many layers as needed to correctly receive the message. Therefore, the stream of packets that are delivered by the multicaster can be adaptive on a per-receiver basis, *i.e.*, the redundancy degree is chosen according to the needs of each destination, so unwanted traffic load can be considerable reduced. So, the “the boat unbalanced by the heaviest” problem, which we have seen in subsection 2.2.1.4 to be a serious issue in an heterogeneous environment, does not occur.

A key aspect to address is the following: how many packets are exchanged within a single layer? Using more and thinner FEC layers, *e.g.*, one packet per layer, allows a finer grain protection and a better adaptation of the traffic load. However, the number of available multicast addresses is limited, so the strategy to better use the multicast address space is to cover more than one packet into a single layer. Specifically, we made the choice of exchanging all the information packets, *i.e.*, k packets that contains portion of the original message, within the first layer. While all the other layers contains a fixed number of packet, and such length is properly chosen by the multicaster considering the redundancy degree to be applied and the available layers.

3.2.3.2 FEC Tuning

When using sender-initiated FEC the usual critics is that centralizing the encoding at the multicaster can overload it if the redundancy degree is too high. Even if layering the FEC technique has implied the advantage of adaptiveness of the redundancy degree with respect to the destination needs, it does not entail any advantage for the multicaster. If there is even a single destination that requires a strong redundancy degree, the multicaster has to provide it at the expenses to be overloaded. We can notice that packet losses into a stub domain are mostly caused by sender and receiver omissions rather than networking omissions. In fact, since into a stub domain there is a single organization in charge of managing the network, it is reasonable to assume that proper policy to minimize the network omissions can be applied. So, when a destination is affected by a severe loss pattern, it is reasonable that it is overloaded. FEC techniques manage losses by pushing more packets towards a destination, while such strategy is effective in case of network omissions, it can be useless, but also harmful, in case of an overloaded node. In fact, the increased number of packets can augment the overloading of the destination rather than mitigate it. For this reason, the multicaster does not set the applied redundancy on the worst case loss patter, but on the 75th percentile, *i.e.*, the highest 25% of the collected loss estimations is cut off and the mean is computed on the obtained data set. In addition, when the destination is aware that it occurred in an overload, it drops the number of layers to whom it is subscribed. However, this way we have a portion of the destinations that do not receive certain messages, for this reason, we use the proactive gossiping presented in subsection 3.2.2.4 to guarantee message delivery also for nodes that experimented losses untreated by the Layered FEC.

Chapter 4

Experimental Evaluation

This chapter evaluates the effectiveness and quality of the strategies presented in the previous chapter in terms of dissemination latency, timeliness and reliability. We have chosen to perform simulation-based experiments using the Omnet++ environment and some of the available tools for the network emulation, topology generation and overlay emulation. Such simulations prove the success of TODAI at satisfy the dissemination challenges imposed by LCCIs and presented in subsection 1.1.3.

4.1 Preliminaries

4.1.1 Selection of the Evaluation Technique

Assessing the quality of a given data dissemination protocols is typically performed using three evaluation techniques: *analytical modeling*, *simulations* or *on-field measurements* through prototypes. There are a number of factors that need to be accurately considered when selecting which of these techniques is the right one to be adopted for a performance evaluation. There are some considerations [152] that may help to take the best decision and are listed in Table 4.1:

1. *Time to achieve the evaluation*: analytical modeling is characterized by extremely low time to obtain any result, while simulations take a longer time. Measurements usually are placed in the middle by taking less time than simulations and more than analytical models. However, measurement campaigns are strongly affected by the Murphy's law,

Table 4.1: Criteria for Selecting a Performance Evaluation Technique

Criterion	Analytical Modeling	Simulation	Measurement
Time required	Small	Medium	Varies
Accuracy	Low	Moderate	Varies
Comprehensiveness	High	Medium	Low
Cost	Small	Medium	High
Saleability	Low	Medium	High
Applicability	Small	Medium/Large	Varies

i.e., if there is anything may go wrong it will, so the time for measurements in practice is the most variable.

2. *Accuracy of the obtained results*: analytical models are built by using so many simplifications and assumptions that their accuracy level is quite low. On the other hand, simulations incorporate more details and are based on less assumptions so that they provide results that are closer to the real one. Measurements may sound the best choice possible to achieve high accuracy in the performance evaluation. However, this may not be true in all the cases since many environmental parameters, *e.g.*, system configurations, workloads, duration of the measurements, may negatively affect the measurement campaign by compromising the veracity of the outcomes. In addition, the parameters adopted in a measurement campaign may not be representative of the real usage of the system under evaluation, leading to inaccurate conclusions.

3. *Comprehensiveness*: the goal of an evaluation is not only to assess the performance of a given system and to compare to several other ones, but also to comprehend the effects of the system parameters on the achievable behavior. In this sense, analytical modeling offers the best insight on the consequences of certain parameter configurations on the

system behavior, while simulations could not clarify well the interdependencies among the different parameters. Last, measurements is the worst technique in this respect, since it is challenging to understand if an improvement in the system performance may be due to a change in the environment or due to a particular parameter setting.

4. *Cost allocated to the evaluation*: measurements represents the most costly among the three techniques since it requires real equipment and it is time-consuming to conduct. On the other hand, analytical modeling require only paper and pencil, so it is the cheapest technique.

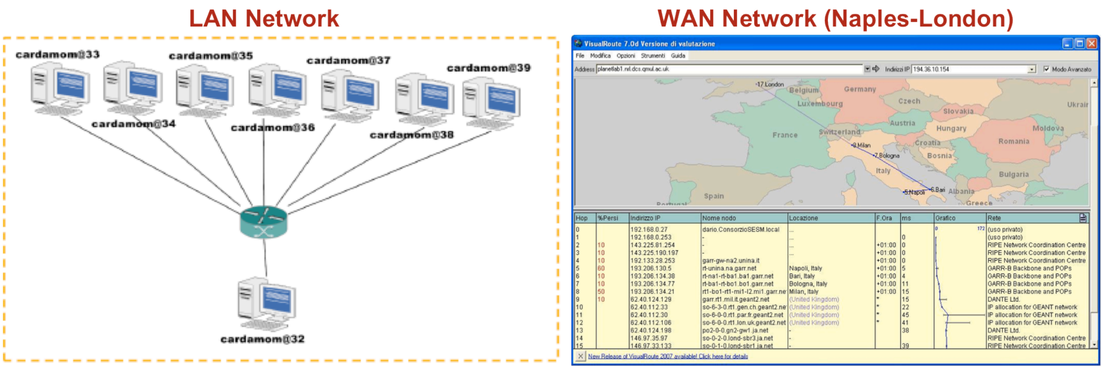
5. *Saleability of the outcomes*: the outcome of a performance evaluations has to be convincible: measurements easily convince others, while results taken from analytical modeling are considered in a more skeptical manner. For this reason, they are teamed up with simulations or measurements to validate the obtained results.

6. *Applicability*: another important factor is the scale of the systems that can be evaluated with a given techniques. Analytical modeling is typically used in the context of small-scale systems, while simulations can be also used for systems of large scale[1]. In the case of measurements, a traditional limit of applicability consisted of the available resources in the laboratory, but in the last years things rapidly changed with novel architectures that allows sharing of computational resources, such as Grid Computing [153], or innovative real testbeds that allow world-wide measurements, such as

[1]The scale of the system that can be simulated depends on the computational power of the used computer, however, techniques of distributed simulation have been recently proposed in order to overcome the limitations imposed by the adopted physical device

PlanetLab [154].

From the previous considerations, especially the last one, it is evident that analytical modeling is not a viable technique to evaluate the performability characteristics exhibited by TODAI. Therefore, the only possibility is to use simulations or measurements, or a combination of them. What is the best choice? To answer this question, we have to consider an important requirements for an high-quality performance evaluation: the outcomes of an evaluation need to be easily reproduced without too much variance (*Repeatability*). Simulations implicitly guarantee repeatability of the achieved resulted, however, the repeatability of on-field measurements strongly depends on the controllability of the experiments, *i.e.*, the behavior on the resources is not to be too much variable and unpredictable, but it has to exhibit a stable behavior. Since measurements of large scale infrastructures such as TODAI require the use of Internet, the question that we have put to ourselves is: are the dynamics of Internet stable and controllable? In literature, it is known that the network behavior of Internet is not stable, but suddenly changes affected by the current traffic load experienced by the overall or a limited part of the network [131]. However, we decided to empirically demonstrate the uncontrollability of communications over Internet by performing a measurement campaign aiming at sssessing the dissemination latency of a DDS-compliant publish/subscribe service by using two different testbed illustrated in Figure 4.1: a path interconnected a pair of nodes over a LAN in the cluster belonging to the CINI-ITEM "Carlo Savy" laboratory at the university of Naples "Federico II" and a network path from a node in Naples to one in London over PlanetLab. We have considered the variability of the


LAN Network
cardamom@33
cardamom@34
cardamom@35
cardamom@36
cardamom@37
cardamom@38
cardamom@39
cardamom@32
WAN Network (Naples-London)
VisualRoute 7.0d Versione di valutazione
File Modifica Opzioni Strumenti Guida
Address planetlab1.rsl.dcs.qmul.ac.uk
Indirizzo IP 194.36.10.154
Modo Avanzato
17 London
8 Milan
7 Bologna
6 Bari
5 Napoli
New Release of VisualRoute 2007 available! Click here for details

Hop	%Persi	Indirizzo IP	Nome nodo	Locazione	F.Ora	ms	Grafico	Rete
0		192.168.0.27	dario.ConsorzioSESM.local	...			0 172	(uso privato)
1		192.168.0.253	-	...		0		(uso privato)
2	10	143.225.81.254	-	...	+01:00	0		RIPE Network Coordination Centre
3	10	143.225.190.197	-	...	+01:00	0		RIPE Network Coordination Centre
4	10	192.133.28.253	garr-gw-na2.unina.it		+01:00	0		RIPE Network Coordination Centre
5	60	193.206.130.5	rt-unina.na.garr.net	Napoli, Italy	+01:00	5		GARR-B Backbone and POPs
6	10	193.206.134.38	rt-na1-rt-ba1.ba1.garr.net	Bari, Italy	+01:00	4		GARR-B Backbone and POPs
7	10	193.206.134.77	rt-ba1-rt-bo1.bo1.garr.net	Bologna, Italy	+01:00	11		GARR-B Backbone and POPs
8	50	193.206.134.21	rt1-bo1-rt1-mi1-l2.mi1.garr.ne	Milan, Italy	+01:00	15		GARR-B Backbone and POPs
9	10	62.40.124.129	garr.rt1.mil.it.geant2.net	(United Kingdom)	*	15		DANTE Ltd.
10		62.40.112.33	so-6-3-0.rt1.gen.ch.geant2.ne	(United Kingdom)	*	22		IP allocation for GEANT network
11		62.40.112.30	so-6-0-0.rt1.par.fr.geant2.net	(United Kingdom)	*	45		IP allocation for GEANT network
12		62.40.112.106	so-6-0-0.rt1.lon.uk.geant2.net	(United Kingdom)	*	41		IP allocation for GEANT network
13		62.40.124.198	po2-0-0.gn2-gw1.ja.net	-		38		DANTE Ltd.
14		146.97.35.97	so-0-2-0.lond-sbr3.ja.net	-				RIPE Network Coordination Centre
15		146.97.33.133	so-0-1-0.lond-sbr1.ja.net	-		39		RIPE Network Coordination Centre

Figure 4.1: Testbed used to evaluate the controllability of Internet-scale measurements

obtained measurement samples as a measure of the controllability of the adopted testbed and formulated such variability as half of the statistic index *Inter-Quartile Distance*, *i.e.*, the difference between the third percentile[2] and the first one[3]. As shown in Figure 4.2, the latencies measured over the LAN path exhibit a restrained variability since the IQR of the collected measures is only equal to $20\mu s$ while the median value of the data set is $3160\mu s$, so all the samples are very close to the median and the error made when approximating them with the median is $10\mu s$ which is almost negligible. On the other hand, the latencies measured over the PlanetLab path exhibit a strong variability since the IQR of the collected measures is equal to $37672, 5\mu s$ while the median value of the data set is $540035\mu s$. By just comparing the IQR of these two experiments, it is evident that Planet is not a controllable environment, so it cannot achieve reproducible results. Such conclusion has been also supported by a paper [155] on how to efficiently perform evaluation in PlanetLab, which states that it has not been designed to perform controlled experiments.

[2]As said in subsection 3.2.3.2, it is computed by cutting off the highest 25% of the given data set and calculating the mean on the obtained data set.

[3]It is computed as the third percentile, however, the highest 75% of the original data set is cut off.

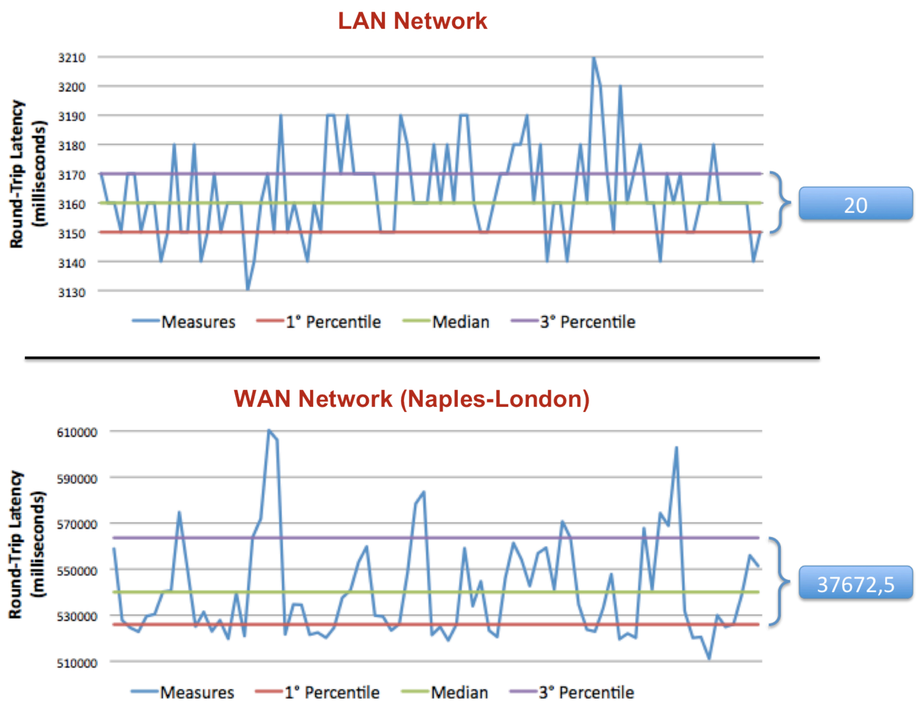


Figure 4.2: Comparison of the variability within the obtained measures

4.1.2 Simulator Selection

Several network simulators have been proposed in the years both by academia and industry and characterized by a free use or a commercial license. In the first case simulators are available as open source applications, so users are able to contribute to them by extending their functionalities, detecting and correcting to their bugs. On the contrary, commercial simulators are not available for free and do not give provide open source code, but usually offer complete and up-to-date documentation that allows quickly learning how to use them. We focused our attentions on the academic simulators due to their flexibility and extensibility. In fact, academic simulators have the advantage to presents several third-party models that a user can use to reduce the time required to realize their simulation models.

Typical academic, open source, simulators include the following ones [156]:

1. *ShawnNet*[4]: it an open-source simulator with an elevate abstraction level, which give the possibility to write algorithms in C++ language and do not present a graphical interface. It is characterized by few third-party models available within the simulator to be imported in new simulation models. Allows to emulate the network conditions using proper communication models while a transmission model allows simulating delivery behaviors. Moreover, it do not provide any automatic means to process the results of a simulation.

2. *Algosensim*[5]: it is an open-source framework for simulating distributed systems by coding in Java. The capability of the simulator to emulate network behaviors is limited, *i.e.*, it can emulate only packets losses, as the number of nodes that can be interconnected. Moreover, the graphical interface of the simulator provide only basic functionalities, but it offers the possibility to statistically process the results of simulations performed by the user.

3. *Smurph*[6]: it is a simulator specialized on emulating communication protocols at the medium access control (MAC) level. It can be viewed as a combination of a protocol specification language based on C++ and an event-driven, discrete-time simulator that provides a virtual (but realistic) and controlled environment for protocol execution.

4. *Network Simulator*[7] (NS): it is one of the most adopted behavioral simulator that

[4]www.swarmnet.de/index.html?lang=en

[5]tcs.unige.ch/doku.php/code/algosensim/overview

[6]www.cs.ualberta.ca/ pawel/SMURPH/smurph.html

[7]www.isi.edu/nsnam/ns/

targets at networking research of every communication protocols at every level of the ISO/OSI stack. Due to its success in the networking community, it is characterized by several models than can be imported in new models to reduce the time needed to realize them. Moreover, models are built according a modular approach, where aspects of a communication protocol is incapsulated in a C++ modules. A graphical user interface is provided in OTCL, which allows writing scripts that can overcome the lack of an automatic tool for results processing. To overcome the limitations of the current version of this simulator, namely ND-2, a new version, called NS-3, has been released with the scope of improve the simulation performance and introduce the possibility of using C++ only to implements simulations models.

5. *Glomosim*[8]: it is a simulator that adopt a multi-processor architecture where a user can decide on which process a specific part of the model can be executed. A simulation model is realized in modules written in Parsec, each one tailoring a specific layer of the communication protocol to simulate. The user is helped in its simulation efforts by an advanced graphical interface, but the network failures, cannot be emulated.

6. *J-Sim*[9]: it is a simulation environment written in Java with a graphical interface in Tcl and is based on a component-oriented approach to model the behavior of nodes in a distributed system. It allows the user to manage the organization of the simulation model in threads, to parallelize the model execution over multiple processors and to

[8]pcl.cs.ucla.edu/projects/glomosim/
[9]nsr.bioeng.washington.edu/jsim/

automatically extract statistic informations from the results collected during experiments. The main disadvantages are its low possibilities to model network failures and the high complexity to implement simulation models due to the low expressivity of the Tcl language that require several commands even to realize simple operations.

7. *Omnet++*[10]: it is a general-purpose modeling framework that is currently heavily used in the simulation of networked systems and distributed algorithms by the academic research. It is characterized by a modular and extensible architecture of C++ modules, with a Tcl graphical interface, recently extended by providing an Eclipse-enhanced IDE. Due to its success, there are several third-party models that featured academic projects made available through the web page of the simulator and users can import in their models. The simulator comes with several analysis tools that allows the user to study the statistical features of the performed experiments.

We have summarized the key features of the previous network simulators in Table 4.2, where it is evident why NS and Omnet++ are the most adopted in the recent research papers on networking and communication protocols and algorithms. Both Omnet++ and NS-3 exhibit similar scalability [157] according to simulation run-time and memory usage for simulations of large-scale systems. We considered the adoption of Omnet++ in our research efforts since NS-3 is still in its early stage and just a few third-party models can be used as off-the-shelf in new simulation models and collections of models for NS-2 still need to be ported in the new version of the simulator.

[10]www.omnetpp.org/

Table 4.2: Characteristics of the current network simulators

	Modeling Language	Simulation Type	Emulation	GUI	Data Processing	Extendibility	Modularity
ShawnNet	C++	Continuous time	No	No	No	Yes	Yes
Algosensim	XML/Java	Continuous time	No	Yes	No	Yes	No
Smurph	C++	Discrete time	Yes	Yes	Yes	Yes	Yes
Network Simulator	Otcl/C++	Discrete time	Yes	Yes	No	Yes	Yes
Glomosim	Parsec	Discrete time	No	Yes	No	Yes	Yes
J-Sim	JACL/Java	Discrete time	No	Yes	No	Yes	Yes
Omnet++	NED/C++	Discrete time	Yes	Yes	Yes	Yes	Yes

4.2 Experiment Setup

We have realized a simulation model of TODAI to evaluated the effectiveness of the techniques presented in chapter 3. Specifically, we have used Omnet++ as simulation frameworks plus some related tools[11], as illustrated in Figure 4.3:

1. INET framework has provided the means to model the networking devices as routers and switches, however, we decided to do not specify the networking failures as parameters of such models. The issue is that the path-by-path characterization is not possible since the on-field measurements of loss patterns are only possible on end-to-end basis (we can detect that a given subscriber lost a message, but not along which path such loss happened), as we have made in the following subsection 4.2.2. Therefore we have inserted a module before the UDP one that drops incoming messages according to the applied loss pattern.

[11]Refer to Appendix B for any detail on Omnet++ and such tools.

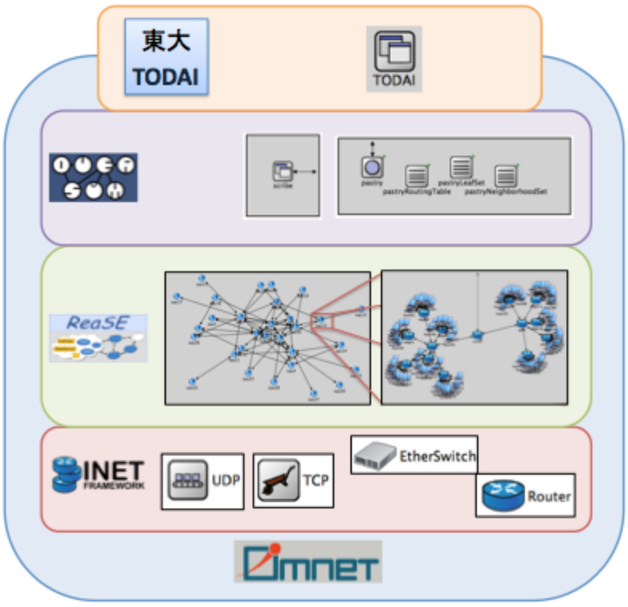


Figure 4.3: Layered implementation of the simulation model

2. Rease has been used to generate the interconnection topology among them and the end hosts as described in the following section 4.2.1;

3. Oversim has been used to incorporate Pastry simulation modules and as basis to implement the strategies presented in section 3.2.

Last, we have applied in our simulations a failure model of the software faults and a workload, as respectively presented in subsection 4.2.3 and in 4.2.4.

4.2.1 Network Topology Generation

The correct model of the Internet topology, *i.e.*, the connectivity graph among its nodes [158], is one of the key aspect to address when aiming at making realistic simulations of systems that use Internet to convey information. As illustrated in subsection 3.1.1, Internet is structures as an interconnection of distinct routing domains, also known in literature

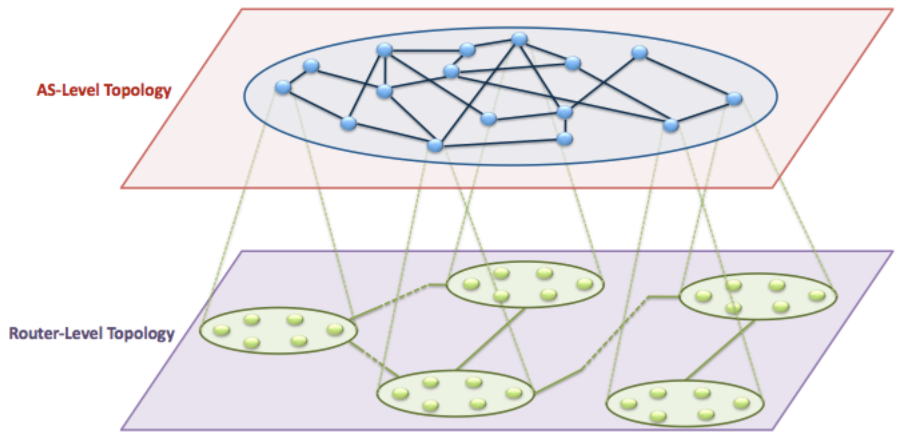


Figure 4.4: Topology model composed by two abstract layers

as *Autonomous Systems* (AS), which adopts an interior gateway protocol to internally route packets while uses an exterior gateway protocol to forward packets towards others ASes [159]. Therefore, the topology of Internet can be characterized using two distinct abstraction levels, as shown in Figure 4.4:

1. *Inter-AS topology*, also called AS-level topology: a node of the connectivity graph at this level of abstraction represents a single AS, while the edges are the BGP peering[12] relations that occur between two ASes;

2. *Intra-AS topology*, also called Router-level topology: at this level of abstraction nodes of the connectivity graph represent end-hosts or hardware devices, while edges are physical connections among them.

Currently, several different models and tools to generate the Inter-AS topology are available, while the same is not true for the Intra-AS topology. This is caused by the lack of data on

[12]Border Gateway Protocol (BGP) is the core routing protocol of the Internet, while BGP peering indicates the connections between BGP speakers, *i.e.*, routers running BGP that have different AS numbers.

the topology and connectivity within ASes. In fact, ISPs are reluctant to make public this information since they do not want to violate the privacy of their clients [160]. However, this issue is quite marginal for the effectiveness of the overall simulation model. In fact, in literature [158] Inter-AS topology is considered more critical since it is relevant to the overall inter-networking infrastructure, while Intra-AS is only relevant to the data distribution within a single AS. If we consider a message that has to be exchanged between two hosts, one in Australia and one in Norway, the transmission quality is strongly influenced by the topological characteristic of the interconnection among AS rater than the topology of the ASes of the two hosts.

4.2.1.1 Inter-AS Topology Modeling

The main models in literature for modeling Inter-AS topology are the following ones:

1. *Waxman* [161]: nodes are placed in a random manner on a two-dimensional plane Π with the maximum euclidian distance among them equal to L. Then, the probability that two nodes of Π, namely u and v are interconnected is expressed as a spectral distribution as follows:

   $$P(u, v) = \alpha e^{-d/\beta L} \tag{4.1}$$

   where $0 < \alpha, \beta \leq 1$ while d is the euclidian distance among the nodes.

2. *Barási-Albert (BA)* [162]: the process of interconnecting different nodes is performed by progressive incrementation, *i.e.*, new nodes are connected to already-existing and well-connected nodes. Starting from a network with m_0 isolated nodes, m new connections, with $m \leq m_0$, are added with probability p. One end of each link is attached

to a random node i, while the other end is attached to a node j selected by preferring the more popular, well-connected, nodes with probability formulated as follows:

$$\Pi(k_i) = \frac{k_i + 1}{jk_j + 1} \tag{4.2}$$

where k_j is the degree of node j. With probability q, m links are rewired and new nodes are added with probability $1-p-q$. A new node m has m new links that, with probability $\Pi(k_i)$, are connected to nodes i already present in the system.

3. *Generalized Linear Preference* (GLP) [163]: It focuses on matching characteristic path length and clustering coefficients by using a probabilistic method for adding nodes and links recursively while preserving selected power law properties. When starting with m_0 links, the probability of adding new links is defined as p, while for each end of each link$\Pi(d_i)$ is the probability of choosing node i, defined as follows:

$$\Pi(k_i) = \frac{d_i - \beta}{\sum_j (d_j - \beta)} \tag{4.3}$$

where $\beta \in (-\infty, 1)$ is a tunable parameter indicating the preference of nodes to connect to existing popular nodes.

4. *Inet* [164]: it produces random networks using a preferential linear weight for the connection probability of nodes after modeling the core of the generated topology as a full mesh network. Inet sets the minimum number of nodes at 3037, the number of ASes on the Internet at the time of Inets development. By default, the fraction of degree 1 nodes α is set to 0.3, based on measurements from Routeviews[13] and

[13]www.routeviews.org/

NLANR[14] BGP table data in 2002.

5. *Positive-Feedback Preference* (PFP) [165]: topology grows by interactive probabilistic addition of new nodes and links. It uses a nonlinear preferential attachment probability when choosing j older nodes for the interactive growth of the network, inserting edges between existing and newly added nodes. The probability to choose a node with degree k is formulated in the following equation:

$$\Pi(k) = \frac{k^{1+\delta ln(k)}}{\sum_j j k_j^{1+\delta ln(k_j)}}, \delta \geq 0 \tag{4.4}$$

Among the previous models, the one that is able to generate the most realistic Inter-AS topology resulted to be the PfP model from a comparison described in [166]. Specifically, the topologies generated by each model have been compared with snapshots of the current Internet obtained by the CAIDA skitter Topology Traces[15] and evaluated according to the following metrics:

- Node Degree Distributions (NDD): the degree of a node is the number of links towards other nodes of the network, while the degree distribution is how the node degree is probabilistically distributed among the nodes of the network;

- Average Neighbor Connectivity (ANC): measure of the mean degree of nodes that are neighbors to a given node;

- Clustering Coefficient (CC): assessment of the degree to which nodes cluster together;

[14]www.nlanr.net/

[15]www.caida.org/tools/measurement/skitter/

- Rich-Club Connectivity (RCC): how well a member of a group "knows" the other members, *e.g.*, if such quantity is equal to 1, then all the members know each other, which in terms of connectivity means that all the nodes are connected each other according to a fully-connected mesh.

Comparative studies [166, 158] have shown that only the PFP model is able to exhibit optimal values in all the previous measures, while the other models are able to guarantee suitable values to only a limited number, as shown in Table 4.3

Table 4.3: Coverage matrix of the connectivity measures by current topology models

	NDD	ANC	CC	RCC
Waxman				
BA	√			
GLP	√			
Inet	√	√		
PFP	√	√	√	√

4.2.1.2 Intra-AS Topology Modeling

In the context of Intra-AS topology, the usual method to build a connectivity graph was the random generation by using variations of the classical Erdös-Rényi random graph [167]. The use of this type of random graph model was later abandoned in favor of models that explicitly introduce non-random structure, particularly hierarchy and locality, as part of the network design [168], since Intra-AS real networks are clearly not random but do exhibit certain obvious hierarchical features. These principles were integrated into the Georgia Tech Internetwork Topology Models (GT-ITM)[16]. However, later on power law relationships in the connectivity of router-level graphs of the Internet were reported by Faloutsos et al. [169].

[16]www.cc.gatech.edu/projects/gtitm/

Since the GT-ITM topology generator fails to produce power law distributions in node degree, it has often been abandoned in favor of new models that explicitly replicate these observed statistics [170]:

1. *Preferential Attachment* (PA) [171] says (*i*) a network is built by the sequential addition of new nodes, and (*ii*) each newly-added node is preferentially connected to some already-existing nodes. So, high-degree nodes are likely to get more and more connections resulting in a power law in the distribution of node degree.

2. *Chung and Lu Method* (CLM) [172] proceeds by first assigning each node its (expected) degree and then probabilistically inserting edges between the nodes according to a probability that is proportional to the product of the degrees of the two given endpoints. If the assigned expected node degree sequence follows a power-law, the generated graphs node degree distribution will exhibit the same power law.

3. *Power Law Random Graph* (PLRG) [173] attempts to replicate a given (power law) degree sequence. This construction involves forming a set L of nodes containing as many distinct copies of a given vertex as the degree of that vertex, choosing a random matching of the the elements of L, and applying a mapping of a given matching into an appropriate (multi)graph.

The work presented in [170] proved, however, that the topologies generated by the previous three models are not representative of the real structure of an AS. On the other hand, it shows that better topologies can be obtained by considering that a reasonably "good" design for a single ISPs network is the one in which the core is constructed as a loose mesh

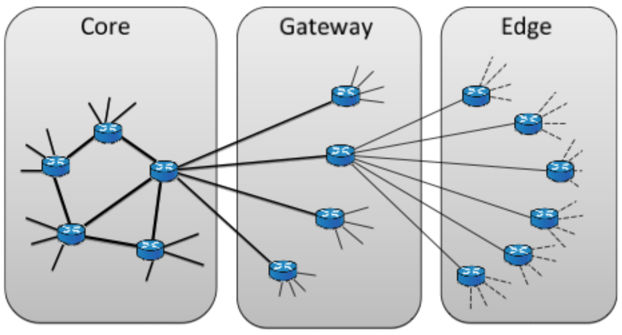


Figure 4.5: Hierarchical organization of the routers within an AS according the HOT model

of high speed, low connectivity, routers which carry heavily aggregated traffic over high bandwidth links. Such innovative generation model, called *Heuristically Optimal Topology* (HOT), articulates the generation efforts in two steps and structure the routers in a three-tier hierarchy, as shown in Figure 4.5:

1. the PA method is applied for generating an initial topology;

2. a heuristic, nonrandom, degree-preserving rewiring of the links and nodes in the initial graph is performed: (*i*) C lower-degree nodes at the center of the graph are assumed as core routers, while the other higher-degree nodes hanging from each core are considered as gateway routers; (*ii*) connections among gateway routers are adjusted so to equally distribute their aggregate bandwidth; and (*iii*) the number of edge routers is arranged according the degree of each gateway.

4.2.2 Network Conditions

After determining the most suitable Internet topology, it is needed to characterize the realistic behavior of the adopted network for the realization of effective simulations for large-scale systems. Many studies have been conducted on measuring latency and packet loss over wide-area networks, however, only few of them agree due to the enormous diversity of Internet [75]. We have chosen to conduct a measurement campaign in order to characterize the features of some European Internet paths, and populate with the achieved features the parameters of our simulation models.

A simple tool has been designed to analyze the latency and packet loss behavior between two hosts on a specific path. This tool consists of two programs: a *traffic generator and analyzer* and a *traffic echoer*. The first program sends UDP packets at a fixed publishing rate toward the second one, which is continuously listening for UDP packet on an apposite port. All the received packets are immediately sent back to the sender, which computes their round-trip latencies. The packets that do not have an estimated latency are considered lost. At the end of a test, the network behavior is described in terms of 1) *Delay* (*i.e.*, round-trip time latencies), 2) *Packet Loss Rate* (PLR) (*i.e.*, percentage of the lost packets), and 3) *Average Burst Length* (ABL) (*i.e.*, number of the packets lost in a row).

Performing a measurement campaign to characterize Internet behavior involves facing several problems:

Testbed Choice : The measurements have to be performed on a testbed that is representative of Internet.

Solution: PlanetLab [154] is a geographically distributed overlay platform designed for deploying and evaluating services over wide-area networks. It has been used for the following reasons:

- Even if many of PlanetLab nodes are connected to Internet2, or more generally, the global research and education network (GREN) [174], this does not highly influence the observable behavior;

- PlanetLab has been designed to subject network services to real-world conditions;
- Applying the best practices presented in [155] allows PlanetLab to be suitable for measurements of network conditions.

Loss Observation : The implemented measurement tool observes the network behavior at the application level, however, there is not-always an one-to-one mapping between messages and packets. If its size is greater than the Maximum Transmission Unit (MTU), a message is fragmented into several packets, and the loss of a packet causes the loss of the entire message. Thus, the message loss rate is always higher than of the packet loss rate caused by the network dynamics. Thus, when message fragmentation is used, observing the message losses is not suitable to achieve the realistic behavior. *Solution*: The message size has been set equal to 1464 bytes, which, adding the size of the headers of the underlying protocols, is equal to the MTU for Ethernet, so IP fragmentation is avoided.

Publishing Rate Dependence : One of the reasons of packet losses is represented by router buffer overflow. Increasing the message publishing rate may overload the routers and affects the packet losses [175].
Solution: The tests are performed using three different publishing rates, *e.g.*, 200, 100 and 50 messages per second. Using the formula proposed in [176] and the outcome

of these three tests allows calculating the conditional loss probability, *i.e.*, the probability that the next packet will be lost known the previous one was lost, at a new publishing rate.

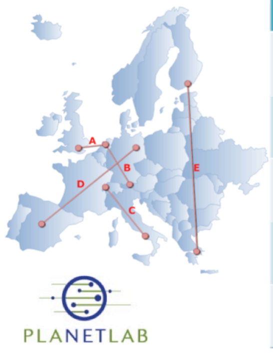
Path Dependence : The type of adopted paths (*e.g.*, path length, type and number of crossed network infrastructures) can affect packets losses.

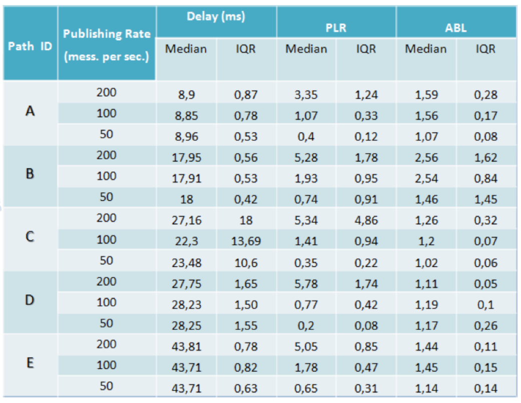
Solution: The measurements are performed on five different paths, as shown in the left side of the Figure 4.6. This path diversity allows achieving a network characterization that is statistically significant.

Outliers Presence : The obtained measures could be affected by outliers, *i.e.*, values out of range respected the others measures. This can distort the statistics on the measurement data set.

Solution: The solution is to use robust statistics [177], *i.e.*, representative of the real trend into the data set and not function of outliers. Thus, the median has been chosen to analyze the central tendency of a data set, and the IQR for the data dispersion.

The table in the right side of Figure 4.6 summarizes the results of one-day round-trip loss collection for each path and publication rate. Overall, the network behavior exposes a not-excessive packet loss rate, however, it is quite bursty, *i.e.*, packet losses are not isolated, but correlated. As expected, the loss rate is directly related with the publishing rate, however, the average burst length does not clearly show this trend, as shown in Figure 4.7. Moreover, the network features depend on the path length, however, even if two path have the same length (*e.g.*, paths B and C), they can not expose the same loss statistics. As also observed





Path ID	Publishing Rate (mess. per sec.)	Delay (ms)		PLR		ABL	
		Median	IQR	Median	IQR	Median	IQR
A	200	8,9	0,87	3,35	1,24	1,59	0,28
	100	8,85	0,78	1,07	0,33	1,56	0,17
	50	8,96	0,53	0,4	0,12	1,07	0,08
B	200	17,95	0,56	5,28	1,78	2,56	1,62
	100	17,91	0,53	1,93	0,95	2,54	0,84
	50	18	0,42	0,74	0,91	1,46	1,45
C	200	27,16	18	5,34	4,86	1,26	0,32
	100	22,3	13,69	1,41	0,94	1,2	0,07
	50	23,48	10,6	0,35	0,22	1,02	0,06
D	200	27,75	1,65	5,78	1,74	1,11	0,05
	100	28,23	1,50	0,77	0,42	1,19	0,1
	50	28,25	1,55	0,2	0,08	1,17	0,26
E	200	43,81	0,78	5,05	0,85	1,44	0,11
	100	43,71	0,82	1,78	0,47	1,45	0,15
	50	43,71	0,63	0,65	0,31	1,14	0,14

Figure 4.6: Paths used to characterize the Internet behavior and summary of the obtained characterizations per each path

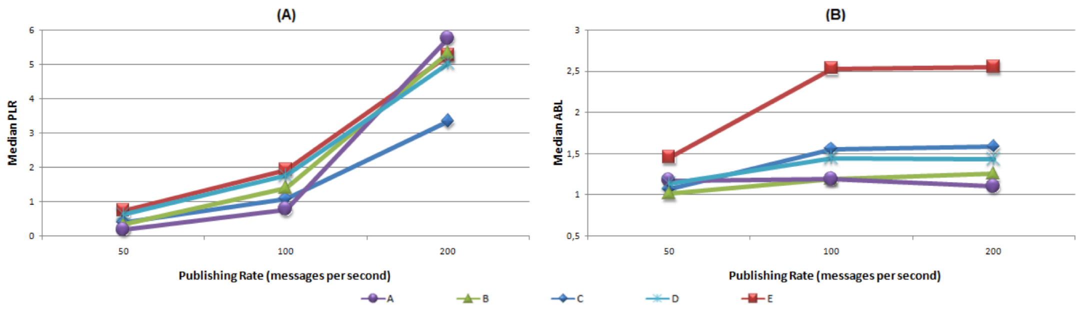


Figure 4.7: Trend of A) PLR and B) ABL over all five paths

in previous works [74, 75], loss statistics are not constant during the day due to the higher amount of traffic, especially HTTP requests, corresponding to intense web surfing activity during specific hours of the day. For example, ABL on the path *B*, *i.e.*, between Innsbruck and Amsterdam, with a publishing rate of 200 messages per second exposes higher values in the first hours of the morning and in the evening, as shown in Figure 4.8. This phenomenon is not-deterministic because it depends on the activity on the network in a specific period of time, which is not replicable, so the loss statistics are highly variable and uncontrollable.

Concluding, we decided to assume the network conditions for our experiments from the

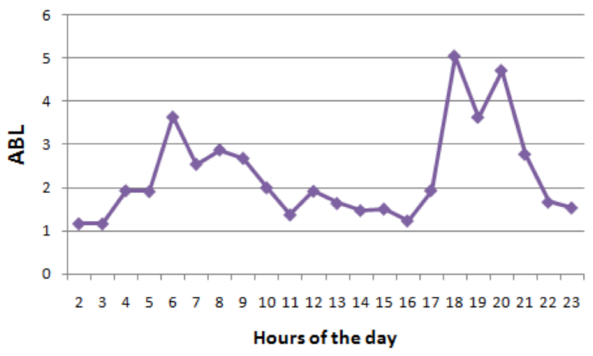


Figure 4.8: Variability of ABL during the day on path B and with 200 messages per second

path C that exhibited an average behavior in terms of delay, PLR and ABL.

4.2.3 Failure Modes

As introduced in subsection 1.3.2, a node can fail by crashing due to software and/or hardware faults. In particular, a node failure can cause value or timing domain failures [178]. Here we assume that a node can fail due to following causes:

- **Hardware Faults**: the correct behavior of the node is compromising by a malfunctioning part of the physical device. In particular, our attention is focused on the intermittent faults since they are the most frequent one which can affect a system [179];
- **Software Aging Faults**: Software aging is a term which refers to a condition in which the state of the software or its environment degrades with time.

Recent studies [180] have shown that such category of failures is the major source of application and system unavailability, therefore we have focused only on such category. The faulty behavior of nodes has been modeled according to [181]: a single node is composed of two main distinct components that affect its availability, *i.e.*, the business application ad

Table 4.4: Parameters and their default values used in the simulations (Time in Seconds)

Parameter	description	default value
λ_N	Rate of the software faults during normal periods	1,10231E-02
T_N	Expected duration of the normal periods	27215640
λ_B	Rate of the software faults during abnormal periods	8,33E-03
T_B	Expected duration of the abnormal periods	360
N_{crash}	Number of software errors to crash of the application or of the OS	10
μ_A,μ_O	Scale parameters of the lognormal distribution of the times to errors of applications and OS	12
σ_A^2,σ_O^2	Shape parameters of the lognormal distribution of the times to errors of applications and OS	4,2 ; 4
OS_Recovery_time	Time needed for OS reboot	300
App_Recovery_time	Time needed for application restart	30

the Operative System (OS). Specifically, we have modeled the occurrence rate of software failures within each of these two component as a Lognormal distributions[17], respectively with scale and shape parameters μ_A, μ_O and σ_A, μ_O. When the number of failures within a component exceeds a give threshold, namely N_{crash}, the node crashes. We have considered a fail-recovery model, where a crashed node is recovered by rebooting in order to return again to be part of TODAI. Therefore, after the expiration of the needed time to recovery, namely *OS_Recovery_time* and *App_Recovery_time* respectively for the OS and the Application, the node is restarted and the time for the next failure is reduced by using a distribution with a mean equal to the original one divided by 2^i [181], where i is the number of failures that affected the node during the simulation. In addition, each component can assume two possible states: normal periods, whose expected duration is indicated by the parameter T_N, where the transient fault occurs with rate N, and of abnormal periods, having expected duration T_B, characterized by a higher rate B. The values of the parameters of the failure models are resumed in Table 4.4.

[17]Refer to [182] for evidence on why lognormal distributions are suitable to model software failure rates.

Table 4.5: Parameters and their default values of the workload used in the simulations

Parameter	description	default value
λ_M	Publishing rate of the messages disseminated along TODAI	100 msg per second
D_M	Message size	100MB
N_C	Maximum number of ACCes interconnected	30
N_N	Maximum number of nodes within an ACC	500

4.2.4 Workload

As stated in subsection 1.1.2, it is our opinion that the novel ATM framework that EuroControl is currently developing represent a representative example of the practical usage of LCCI in a real case study. For this reason, we have drawn from this application domain the workload that we have used in our experiments. Specifically, the simulations have been executed considering the requirements of *CoFlight*[18], a novel flight data processing system under development by a collaboration of Thales and Selex-Si. Coflight is an application running in an ATC system to update flight data plans with information received from the RADAR and other air traffic monitoring instruments, and to distribute them to the system tower control of its ACCes ad also towards other ACCes. The requirements of this product are resumed in Table 4.5.

4.3 Empirical Results

In this section we present the outcome of several experiments that we have performed using Omnet++ to evaluate the effectiveness of the reliability means that we have presented in section 3.2. Specifically, (*i*) in subsection 4.3.1 we have investigated the efficiency of the adopted replication technique to tolerate coordinator crashes, (*ii*) in subsection 4.3.2 we

[18]http://www.eurocontrol.int/ses/gallery/content/public/docs/pdf/ses/fab_wks_040928/Technical%20Co-Flight.pdf.

show how the combination of multiple trees and distributed FEC allows achieving strong reliability levels without leading to severe performance fluctuations, last, (*iii*) subsection 4.3.3 revels how a layered FEC is adaptive with respect to the demands of each single destination.

4.3.1 Cluster Availability

The first experiments aim at analyzing the trend of the cluster availability, *i.e.*, the capacity of a message generated within a cluster to leave it and reach other distinct clusters. The first simulation studies such availability as function of the *backup degree* (*i.e.*, the number of replicas when there is only one coordinator within the cluster, or in other worlds how to tune the passive replication) and the publishing rate. Obtained results are reported in Figure 4.9, and show how the cluster availability[19] increases as the number of replicas grows: with the constant publishing rate of $0.1s$, using one replica we have an availability of three nines, with five replicas, we obtain five nines, while with fifteen, seven nines. It is worth noting that the availability improvement is remarkable when we pass from one to five replicas for a cluster. After five replicas, the addiction of further replicas does not lead to significant improvement of the availability level. This means that an high availability level can be achieved with a relatively low amount of replicas in the group, and thus, with an acceptable performance penalty to make such replicas consistent. From the Figure 4.9, it is evident that, decreasing the publishing rate, the availability level increases. The reason of this improvement is that decreasing the publishing rate the *Mean Time To Repair* (MTTR) decreases (in the case of a fault, the Super-Peer have to wait less until the reception of valid

[19]Cluster availability corresponds to coordinator availability, so in this section are interchangeably used.

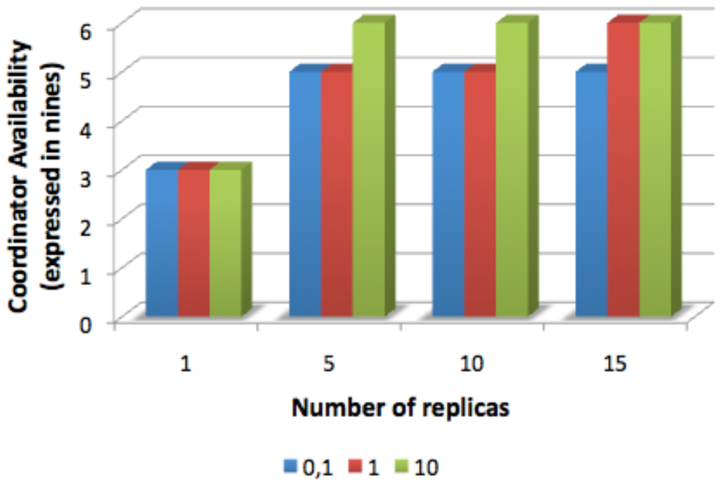


Figure 4.9: Cluster availability (expressed as number of nines) as function of the number of nodes and the publishing rate

data), consequently the system availability[20] increases.

The goal of the second simulation is to study the trend of system availability as a function of the number of coordinators in the cluster. Simulation parameters are five replicas into the local group and a publishing rate of 1 second. The obtained results are shown in Figure 4.10. Doubling from one coordinator to two, the availability improves from five nines to seven and, from this point on, it is constant. Comparing this result with the previous one, it is clear that adding one more coordinator in a cluster results to provide the same availability level achievable in the case with only one coordinator and 10 replica nodes.

[20]The traditional definition of availability is $A = \frac{MTTF}{MTTF+MTTR}$, where MTTF is the Mean Time To Failure, while MTTR the Mean Time To Repair.

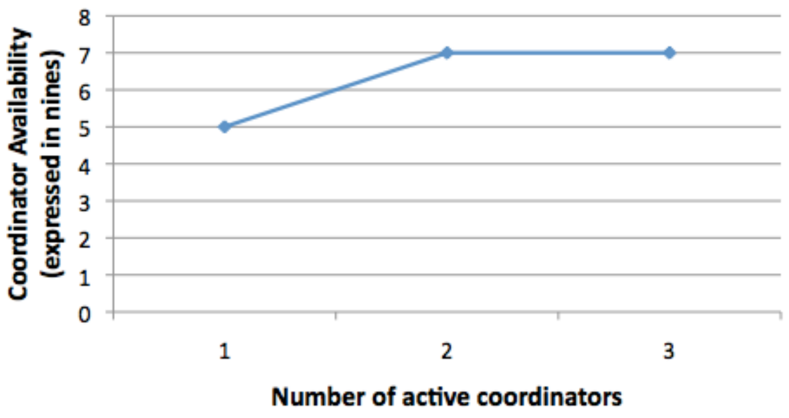


Figure 4.10: Cluster availability (expressed as number of nines) as function of actively replicated coordinators

4.3.2 Effective Internet-scale Dissemination

The scope of the second experiments is to study the quality in terms of reliability and timeliness of the dissemination methods proposed in subsection 3.2.2. Specifically, in subsection 4.3.2.1, the efficiency of the distributed FEC illustrated in subsection 3.2.2.1 is assessed. Then, in subsection 4.3.2.2 the multiple tree-based dissemination stratedy proposed in subsection 3.2.2.2 is evaluated. While in subsection 4.3.2.3, the performance of the combination of the previous two proactive techniques and the reactive RLC gossiping, described respectively in subsection 3.2.2.3 and subsection 3.2.2.4, is investigated.

4.3.2.1 Distributed FEC

The first result that we describe aims at comparing the performance of the basic Scribe implementation, which is available in the OverSim library, and the one we have modified introducing our distributed FEC approach as the resilient mean to tolerate network omissions, namely NE-FEC Scribe. During our simulations, we have measured the average

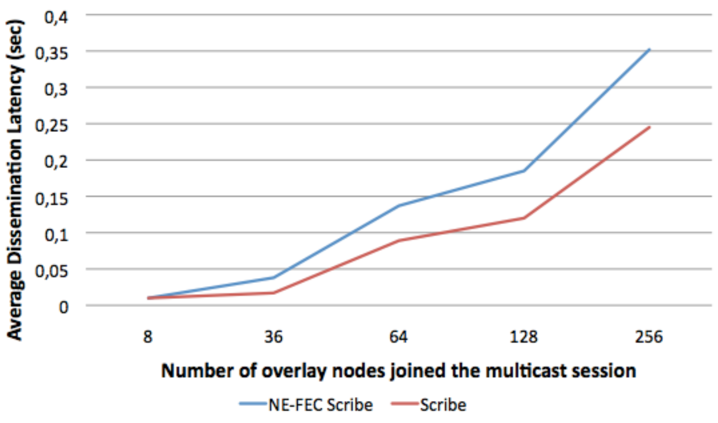


Figure 4.11: Comparison of the performance, in terms of average dissemination latency, between the Scribe implementation of OverSim and the Scribe implementation extended with the Network Embedded-FEC, varying the number of nodes that join a multicast sessions and compose the tree

dissemination latency, *i.e.*, the average of all measures collected at all nodes that register the time to disseminate a message from the tree root to a given node. As shown in Figure 4.11, NE-FEC Scribe exposes higher latency, that increases about 40% respect the latency values exhibits by Scribe. This performance worsening is not surprising, since we have to pay the overhead of performing coding/decoding operations. Figure 4.12 shows the dissemination latencies at each node during a test with 32 nodes that joined the multicast session. In the figure it can be seen that the time to configure the NE-FEC is around 1 second (also in other tests we have registered times to configure NE-FEC almost around 1 second). Another important consideration to make is that the dissemination latency strongly depends on the distance, defined as the number of interior nodes, to the tree root. In fact, Figure 4.12 presents the disseminations latencies arranged in different layers: the light blue lines is the trend of the tree root node, while the first set of lines around the value of $0,035 sec$ are the children of the tree root, then the grandchildren and so on. During such experiments we

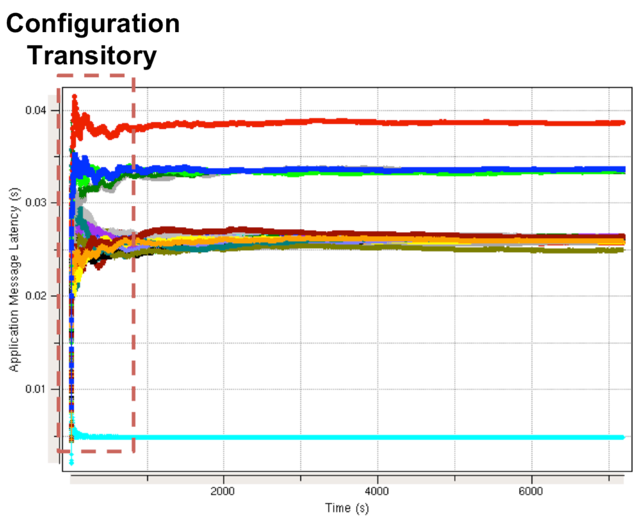


Figure 4.12: Dissemination latency at each node of the multicast tree composed of 32 nodes

have also computed the success rate of the two protocols, *i.e.*, the ratio of averaged delivered messages at each subscriber and the total number of published messages. The scope of measuring the success rate is to evaluate the resiliency. As shown in Figure 4.13, NE-FEC allows delivering almost all the published messages to all the interested subscribers, however, each message is delivered only once. Unfortunately, success rate is not equal to 1, even if only a marginal part of the published messages is not correctly delivered, with a delivery error always lower to 7% of lost messages. On the contrary, during our tests Scribe exhibits a reliability degree around 60% of correctly-received messages. Therefore, NE-FEC allows to achieve a resilient data dissemination. We have further studied the performed simulations in order to assess the timeliness of our approach. We have defined the standard deviation of the dissemination latency at each node as a measure of timeliness. Figure 4.14 shows such timeliness measure of the test with 32 nodes, but also the other tests exhibit similar results. Despite the introduced resilient techniques, NE-FEC Scribe exhibits standard deviations

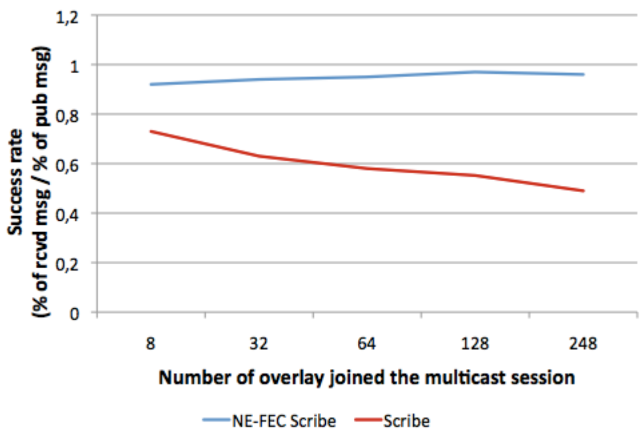


Figure 4.13: Comparison of the success rate, in terms of ratio of received messages and published ones, between the Scribe implementation of OverSim and the Scribe implementation extended with the Network Embedded-FEC, varying the number of nodes that join a multicast sessions and compose the tree

closer to the ones of Scribe. This means that NE-FEC Scribe allows to have performance that are not affected by the network omissions, therefore it is timely.

4.3.2.2 Multiple trees

Figure 4.15 illustrates the results of the tests we have performed to assess the scalability of our multiple-tree approach, indicated in the figure as ”Mod. Scribe”, compared to the unmodified version of Scribe. As shown in Figure 4.15(a), clustering AS-related nodes improves the scalability of the approach, and the latency is affected only by the number of groups. However, if we vary the number of nodes and leave the same number of groups, the trend of the latency is almost constant. We have studied the timeliness quality of our approach in terms of the standard deviation of the delivery time, illustrated in Figure 4.15(b).

With few nodes, the original Scribe exhibits better values for timeliness, but increasing the number of nodes involves a rise in the standard deviation indicating a jitter. On the contrary, the timeliness achieved by our approach exhibits a trend with lower increasing

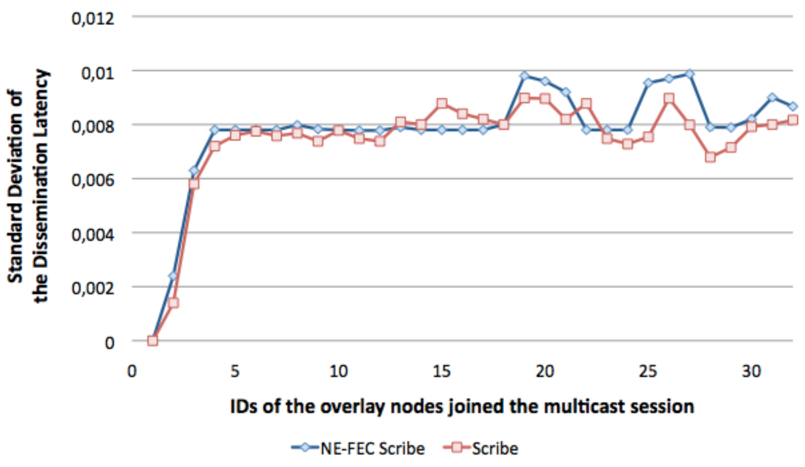


Figure 4.14: Trend of standard deviation the dissemination latency at each nodes of a test with 32 nodes joined the multicast session in case of Scribe and NE-FEC Scribe

rate. The greater standard deviation measured when using our approach than when using Scribe is caused by the use, in our approach, of alternative paths to deliver a message when a path crashes. In fact, as we previously stated, we do not experience the same delivery time of a message though different trees. Thus, the faster path may crash, hence the node receives messages over the slower path. This leads to the variation in the delivery time that motivates the worsening observed in Figure 4.15(b).

We have evaluated the improvement in terms of reliability, measured as the mean success rate, *i.e.*, the ratio of the received messages and the total published messages, shown in Figure 4.15(c). Scribe provides no assurances on reliable event dissemination, so its success rate is low and related to the number of nodes in the tree: increasing the number of nodes causes a drop in the success rate. Our approach on the other hand exhibits better reliability degree, which is not dependent on the total number of nodes. However, the achieved reliability is not equal to 1. In fact we have registered during our tests that about

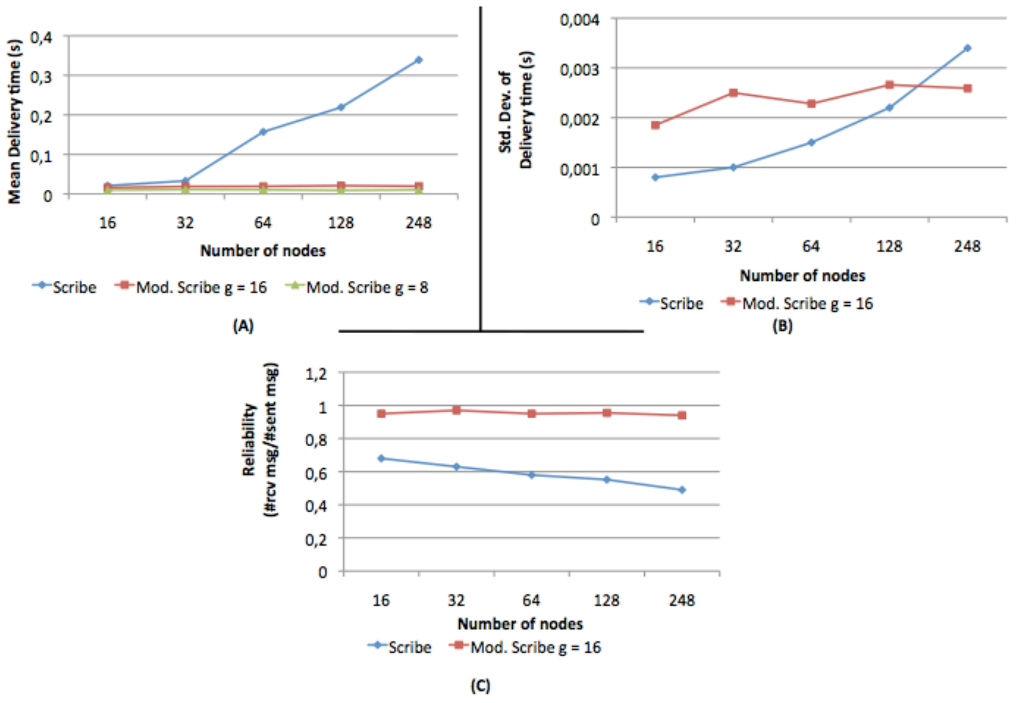


Figure 4.15: Results of the simulation study on the multiple tree approach: a) Scalability, b) Timeliness, and c) Reliability

5% of the published messages do not reach a subset of all the subscribers.

In fact, as we previously stated, it is possible that the minimization performed when a coordinator joins the multicast tree does not return 0 as the diversity measure. The reason for this result is that the achievable diversity is lower bounded by the redundancy degree in the Internet topology, *i.e.*, greater is the path redundancy among the ASes, the closer we can get to the global diversity in the tree forest. In fact, an AS may be connected to only one transit domain, and this limits the achievable diversity, causing subscribers on these ASes to lose messages. In fact, we have made tests with topologies where we forced the ASes to have connections to more than one transit domain, and in these cases reliability is equal to 1.

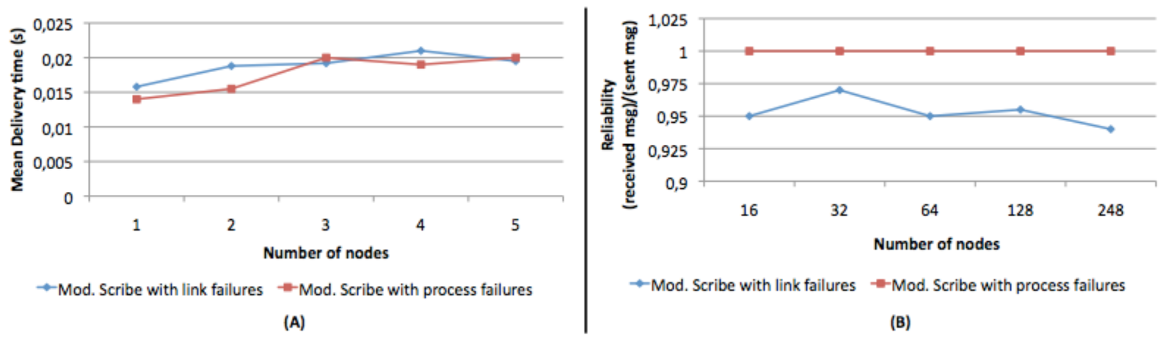


Figure 4.16: Comparison of performance (a) and reliability (b) varying the applied fault model

Lastly, the previously-presented tests have been performed applying only link crashes, and we have seen how they influence delivery time and number of lost messages. To study the effects of process crashes, we have performed the same tests applying only process crashes, then we have compared the obtained results to the results of previous tests. In these tests with process crashes our approach presents comparable delivery latencies to the previous tests, as shown in Figure 4.16(a). Each coordinator receives a copy of the message from a different path characterized by a certain latency. When a coordinator with the best path, *i.e.*, the first to receive the message, crashes, the nodes of the cluster receive the message from the coordinator of the worst quality. This is similar to the case when the path of best quality crashes and the message is received from the path of worst quality. For this reason the performance of our approach when process crashes are applied are similar to the performance when only link crashes are applied. On the other hand, even if process crashes happened, we observed that all the published messages are delivered to all the subscribers, as illustrate in Figure 4.16(b). This means that the replication of the coordinator allows masking any possible process crashes without leading to considerable performance drops.

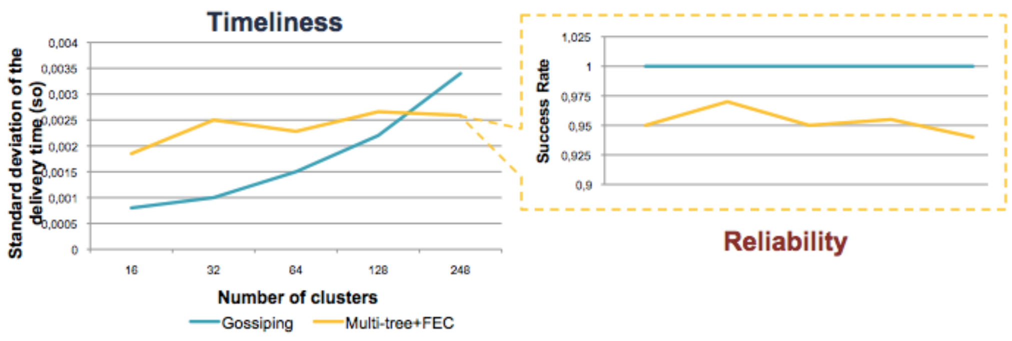


Figure 4.17: Timeliness and reliability of the combination of the proactive techniques

4.3.2.3 Combination of distributed FEC and multiple trees teamed up with gossiping

In this third experiments on the Internet-scale data dissemination, our scope is to evaluate the quality in terms of timeliness and reliability of the combination of the proactive techniques proposed within this dissertation and how data distribution is affected by introducing also a gossiping technique. In the first case, as shown in Figure 4.17, timeliness is not negatively affected, since it exhibit a similar trend observed in the previous sections, however, the reliability is not complete even if it stands really close to delivery all the published messages. On the other hand, when gossiping is also used as a backup technique, timeliness starts to slightly diverge, but, as illustrated in the right part of Figure 4.18, TODAI could achieve the correct delivery of all the published messages.

4.3.3 Effective Cluster-level Dissemination

We conclude these experimental performance evaluation by analyzing the efficiency of the Layered FEC that we have proposed in subsection 3.2.3 by comparing latency, timeliness and reliability to the IP Multicast. Our scope is to show that even if the approach is able to

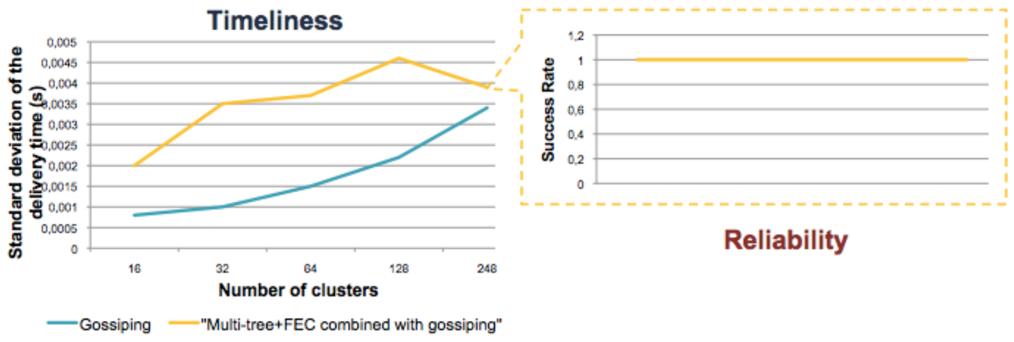


Figure 4.18: Timeliness and reliability of the combination of the proactive techniques teamed up with the gossiping

deliver all the generated messages to each of the interested nodes, we are able to lower the performance overhead and its fluctuations are limited. As shown in Figure 4.19(a), latency is greater than the one exhibited when using only IP MUlticast, due to the use of coding and retransmissions, however, as illustrated in Figure 4.19(b), the timeliness maintains the same trend of IP Multicast. In fact, we have a restrained number of the spikes in the measures, which represents evidences of performance fluctuations and are related to the gossiping. Despite the good timeliness exhibited by the protocol, the reliability level is the desired one, in fact, 4.19(c) shows that all the messages have reached their destinations without the occurrence of any losses.

4.4 Outcomes Discussion

LCCIs require that the adopted dissemination infrastructure satisfy both reliability and timeliness requirements, as illustrated in subsection 1.1.3. Specifically, in the context of publish/subscribe services, reliability is defined as the ability of the service to deliver all the generated messages at ones to all the interested subscribers, as seen in subsection 1.3.1.

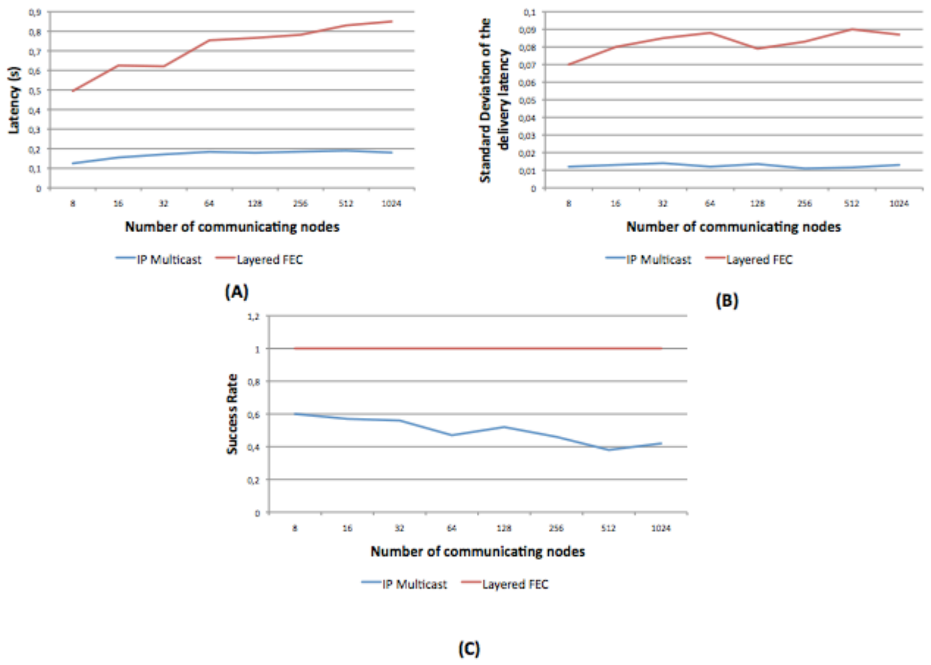


Figure 4.19: Dissemination Latency, timeliness and reliability comparison of the best-effort IP Multicast and Layered FEC

TODAI, as proved by the experimental results illustrated in the previous section, is able to guarantee such reliability property both in the context of a stub domains and in the hostile environment of Internet by tolerating losses, node and link crashes. Such good reliability level, however is not obtained at the expenses of worsening timeliness, as happen in most of the reliability means, as described in section 2.3.

Conclusions

This dissertation addressed the issues of providing reliability and timeliness for data distribution among geographically-distributed nodes in the context of the novel generation of critical MCSes called LCCIs, which adopt a federated architecture. We have proposed a two-tier organization of the adopted data dissemination infrastructure as a super-peer architecture. This choice allows partitioning the problems related to the reliable and timely delivery of messages among the processes composing an LCCI in several sub-problems that can be separately addressed. Specifically, processes are grouped in clusters, each one characterized by specific network conditions, while the clusters are interconnected by Internet connections. To tolerate network failures and link crashes without leading to performance worsening, such dissertation introduce a series of proactive techniques both at the cluster and at the Internet level of the organization. Specifically, a combination of proactive and reactive techniques is proposed in order to have the best of both worlds: the high reliability assurances of reactive techniques and the optimal timeliness of the proactive methods. Last, replication is adopted in order to tolerate any possible crash of cluster coordinators that can cause disconnections within the system.

Experimental simulations have been conducted in order to prove the effectiveness of the proposed approach. In fact, experiment outcomes presented in this dissertation show that the presented strategies achieve the goal of providing jointly reliability and timeliness in the communications within an LCCI despite of the occurrence of crashes and omissions. Such results encourage us to continue our research by implementing the proposed techniques in a prototype that can be used to architect a real LCCI. Specifically, we will devise TODAI as a communication protocol that can be plugged in the context of publish/subscribe solutions that are compliant to the OMG standard called Data Distribution Service. This choice is motivated by the increasing success of such standard in critical scenarios, so that our prototype may found higher interest in the research community and industries involved in developing LCCIs.

Appendix A

Error Correction Coding

Reliable multicasting, i.e., *tolerating the different kind of errors imposed by the physical medium over which data is exchanged, thriving research field in recent years. Techniques to ensure that data is transmitted without errors are based on the use of redundancy. Specifically, there is a rich literature on using spatial redundancy, where additional information is generated by using a certain coding technique and forwarded along with the data information. Due to its intrinsic proactive nature, such approach is called Forward Error Correction (FEC), which is in contrast to techniques with a "backward" perspective using retransmissions.This appendix aims to discuss the general ideas behind FEC solutions and present the different coding techniques proposed in recent years, outlining strengths and weaknesses.*

A.1 Foundations of coding

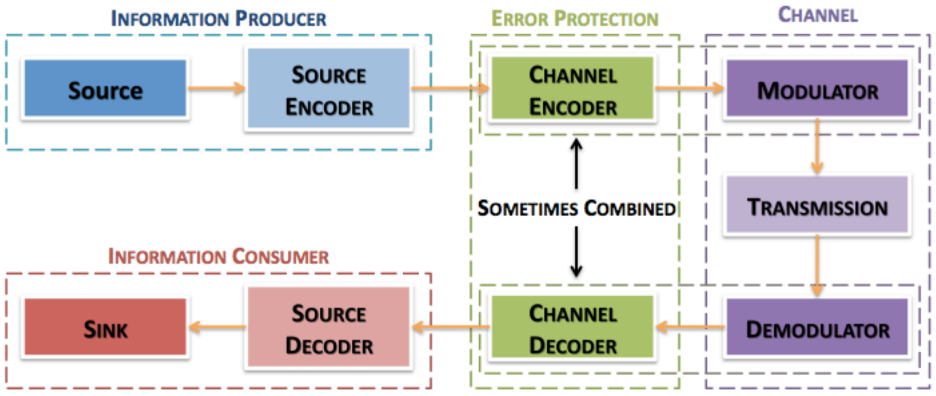


Figure A.1: Overview of a communication system

A communication system is composed by three key elements: a producer that generates information data and disseminates it to a consumer over a certain channel interconnecting the two of them. As shown in Figure A.1, there are many kind of codes employed in such systems [60]:

- A producer is composed of a source of information data, such as a computer file or a video sequence, and a source encoder that codifies information in a digital representation. The couple of these elements are information-theoretically considered as generators of random numbers governed by a certain probability distribution. In addition, the source encoder may apply additional functionalities such as compression [183] to reduce the redundancy in the information stream leaving unchanged its information entropy, or encryption [184] to hide the produced information to unauthorized listeners and provide a secure communication.
- A channel encoder adds redundant information to the data stream received from the producer in order to tolerate errors that may occur during the transmission of the stream over the channel.
- The modulator transforms the bit stream received from the channel encoder into signals appropriate for the transmission over the channel. Sometimes channel encoder and modulator can be combined, *e.g.*, Trellis Modulation (TCM) [185].
- Information is conveyed by a physical medium, *e.g.*, wireless channels or optical fibers.
- The demodulator, channel decoder and source decoder perform the dual operations of what done on the way from the source to the channel, so that the sink can retrieve the original data generated by the source.

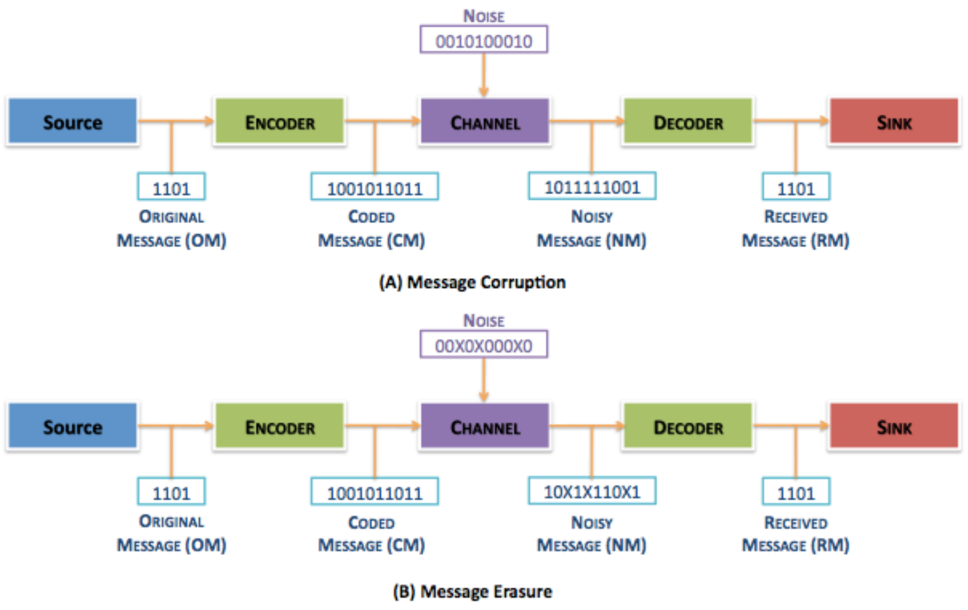


Figure A.2: EC problems: A) message corruption and B) message erasure (in figure, the *X* character indicates the erasure of a digit

Channel encoding is the first stage, while channel decoding is the second of the so-called *Error Correcting* (EC) [186], which aims to guarantee a reliable data communication despite of errors introduced by the channel (*i.e.*, the received message has to be equal to the original message). Specifically, EC deals with the two kind of problems represented in Figure A.2:

1. *Corruption Correction*: the union of modulation, trasmission and demodulation is modeled as a *Binary Symmetric Channel* (BCN), which adds a noise to the data streams exchanged along the channel, so that the Noisy Message is not equal to the Coded Message. The Channel Decoder has the responsibility to detect when and where a corruption took place during the transmission and to properly correct it so that the Received Message corresponds to the Original Message.

2. *Erasure Correction*: the union of modulation, transmission and demodulation is modeled as a *Binary Erasure Channel* (BEN), where erasures represent missing data in a packet or missed packets in a stream. The duty of the channel decoder is providing the original message even if the channel erased a portion of the encoded message.

The focus of this appendix is on the erasure correction, which is realized using the so called *Erasure Codes.* The idea behind the erasure codes is that the encoder accepts in input a block of k symbols, *i.e.*, a sequence of bits with a given length L, and produces a block of n, with $n > k$, symbols called *codeword*, so that any subset of around k encoded symbols are enough to reconstruct the original k symbols of the dataword. Such a code is denoted as (n, k) code and allows a destination to successfully receive a given message even if around $n - k$ losses happened when the message has been exchanged along the channel. Moreover, if the k original symbols are embedded in the codeword, the code is defined *systematic.*

Given a (n, k) code, its *Rate*, namely R, is defined as the ratio of k and n:

$$R = k/n \tag{A.1}$$

so that $R < 1$. Moreover, associated with each channel there is a quantity called *Capacity*, namely C, that represents how much information can be reliably delivered along it. When a channel is affected by errors, the reliability of the delivery process is governed by the *Shannon's channel coding theorem* [187]: "for a transmission rate of a producer that is less than the channel capacity, *i.e.*, $R < C$, there exists a code of a proper length n such that the maximal probability of block error can be made arbitrary small". In its studies Shannon also defined the optimality of a code in terms of how many errors it can detect and then correct. In fact, based on the concept of *Hamming distance* between two codewords[1], it is possible to state that a code with a minimum distance d_{min} allows reconstructing the original message from the received coded messages when the channel imposed $d - 1$ or fewer erasures. An erasure codes is defined optimal if and only if it can recover the original message from any set of k symbols of a codeword, *i.e.*, if $d-1 = n-k$. The codes that verify such optimal property are called *Minimum-Distance Separable* (MDS) codes [188], and are optimal in the sense that they tolerate the largest number of erasures respect to other codes with the same size. In the rest of this chapter we overview several coding techniques that have been proposed during the years to implement MDS codes and applied in several application domains to achieve reliable communications in wired and wireless networks:

- the most adopted codes are the ones indicated as Reed-Solom codes from the names of their inventors, which are discussed in section A.2;
- since the previous codes are characterized by high coding overhead, more light-weigh codes was proposed, based on the studies of Gallager [189]; we illustrate such codes in section A.3, explaining in section A.3.2 a particular sub-class called Turbo Codes;
- last, Luby proposed rateless codes that are able to continuously generate encoded data [99], we focus on such codes in section A.4 by discussing of two different coding solutions to approximate a rateless code.

[1]The Hamming distance between a sequence $\mathbf{x} = [x_1, x_2, ...x_n]$ and a sequence $\mathbf{y} = [y_1, y_2, ...y_n]$ is the number of positions where the corresponding elements differ:

$$d_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} [x_i \neq y_i]. \tag{A.2}$$

We conclude this appendix by comparing the quality of these codes in section A.5.

A.2 Reed-Solomon (RS) Codes

A class of erasure codes, interesting due to their implementation simplicity, are the *Linear Codes*, so called since encoding and decoding operations are equivalent to solving systems of linear equations. Let $\mathbf{x} = [x_1, x_2, ...x_k]$ be a dataword, G a proper $n\ x\ k$ matrix[2], then a (n, k) linear code is represented as follows: $\mathbf{y} = G \cdot \mathbf{x}$, where $\mathbf{y} = [y_1, y_2, ...y_n]$ is a codeword. Assuming that the receiver is reached by $m = n - l$, where l is the number of erasures, the dataword can be reconstructed resolving the following equation: $\mathbf{x} = H \cdot \mathbf{y}$, where H is the inverse of the matrix extracted from the G matrix by removing the rows related to the elements in $\mathbf{y}$ lost by the channel. The decoding process is performable only if the obtained equation presents linearly-independent rows and the matrix extracted by G is invertible.

Linear codes exhibit a decoding process that is resolvable in polynomial time by using Gaussian elimination [60], however, such performance is too high especially for codes of high size. Another source of overhead is the precision used for the computations: if each component x_i of the dataword $\mathbf{x}$ is represented using b bits, and each coefficient $g_{i,j}$ of matrix G is represented on b' bits, then each component y_i of the codeword $\mathbf{y}$ needs $b+b'+[log_2 k]$ bits to be represented without loss of precision [190]. A particular class of linear codes invented by Reed and Solomon in 1960, and for this reason called **Reed-Solomon** (RS) codes [60] (RS), deals with such data expansion in the generated codewords by performing the coding operations over finite fields or *Galois Field* $GF(2^L)$ (so named in honor of Évariste Galois),

[2]The code is systematic if the first k raws of G form the Identity matrix I_k: $G = \begin{bmatrix} I_k \\ NI_{n-k} \end{bmatrix}$.

which is a field[3] that only contains a finite number of elements, which are represented as integer belonging to $[0, ..., 2^L - 1]$ interval. The closure condition of the Galois Fields allows making exact computations on finite elements without requiring more bits in the codeword than the ones in the dataword, and therefore, provides a solution to the data expansion. A (n, k, L) RS code is performed by segmenting the information in k symbols of L bits and by resolving the system $\mathbf{y} = G \cdot \mathbf{x}$ where all the operations are performed in the Galois Field $GF(2^L)$. On the other hand, the decoding process is performed as previously-described for a generic linear code by resolving $\mathbf{x} = H \cdot \mathbf{y}$ in the Galois Field $GF(2^L)$. Moreover in literature, RS codes have been constructed by using two different type of matrix as the not-identical part of the matrix G, *i.e.*, NI_{n-k}:

Vandermonde Matrix , where its element $g_{i,j}$ is the terms of a geometric progression, expressed as $g_{i,j} = i^{j-1}$. Such matrix presents an inversion complexity equals to $O(l \cdot n^2)$, where l is the number of erasures. Moreover, the achievable maximum number of independent raws is equal to the size of the Galois Field, therefore, the following property affects the RS code built with such a matrix: $n + k < 2^L$.

Chaucy Matrix , which is constructed using two distinct set from the Galois Field $GF(2^L)$, *i.e.*, $X = \{x_1, x_2, ...x_k\}$ and $Y = \{y_1, y_2, ...y_n\}$ that $X \cap Y = \varnothing$, and each element of the matrix is defined as: $g_{i,j} = \frac{1}{x_i \oplus y_j}$. Such matrix is invertible with a complexity of $O(l^2)$, where l is the number of erasures, while the code is subject to the following condition: $n + k \leq 2^L$.

[3]In abstract algebra, a set is defined as field only if the result of applying an addition or multiplication on two of its elements still belongs to the set (*closure condition*).

A.3 Low-Density Parity-Check (LDPC) Codes

A.3.1 Generalities

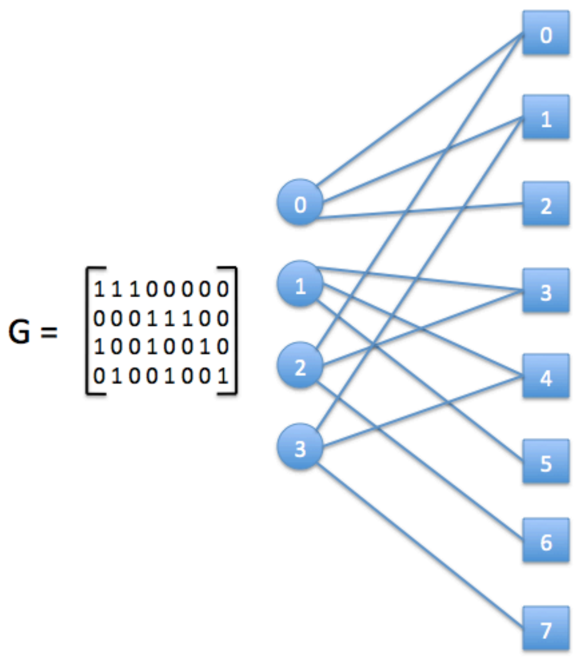


Figure A.3: Generator Matrix G of an LDPC code and its Tanner graph

Low-Density Parity-Check (LPDC) codes were originally proposed in 1962 by Gallager [189], therefore are often called Gallager codes. As stated in their name, they consist of linear codes that exhibit a generator matrix G that is a sparse (which provides the low-density property of the code) parity-check matrix[4]. Due to the sparse nature of its generator matrix, LDPC codes are more often represented using a notation that uses a *Tanner graph*, which is a bipartite graph that connects subcode nodes on the left side to digit nodes on the right side, as shown in Figure A.3. Specifically, the subcode nodes denote rows of the

[4]A parity-check, called C, of a given code is the generator matrix of its dual code named $C^{\perp}$, *i.e.*, the code that generates codewords that are orthogonal to the codewords in C. If we denote with G the generator matrix of C and with H the one of $C^{\perp}$, $C^{\perp}$ is dual to C if and only if $G \cdot H^{\perp} = 0$.

code generator matrix, the digit nodes represent its columns, while an edge connecting a subcode node to a digit node indicates a nonzero entry exists in the intersection of the corresponding row and column. The parity check matrix is generated at random so to reach the Shannon limit for reliable transmission. which is equal to the channel capacity[5]. When the Tanner graph is defined, the subcode nodes hold the symbols of the dataword while the digit nodes compute the symbols of the codeword, as the exclusive-or operation of the symbols contained in the subcode nodes directly connected to the given digit node. Decoding is realized borrowing the belief propagation from the field of Artificial Intelligence to calculate marginals in Bayes Nets [191], which exhibits a linear computational complexity. Experimental studies have shown that the decoding complexity of LDPC codes outperforms the RS ones since it depends linearly, rather than quadratically, to the number of applied redundancy n.The reason of that is explained by looking at how symbols of the codewords are generated: with RS codes every symbol depends on all the symbols in the data word, on the other hand, due to the the sparse Tanner graph as their basis LDPC codes obtain symbol of the codewords only on the base of few symbols of the dataword. This allows fast encoding and decoding operations, however, such advantage is obtained at the cost of a less erasure correction power. In fact, LDPC require more than k symbols to correctly reconstruct the original dataword.

[5]Elias has proved in [186] that random linear codes closely approach the Shannon limit on rate transmission for erasure channels.

A.3.2 Turbo Codes

For a curious twist of history, LDPC codes have been unnoticed for almost thirty years since their invention, overshadowed by the less efficient RS codes. A breakthrough in this area was the development of **Tornado Codes** [188], a class of LDPC specifically designed for erasures. Such codes are constructed using a sequence, with length β of irregular spare graphs as shown in Figure A.4, by using a recursive approach that compute the outcome of the i-th layer of the cascade using the results of the (i-1)-th layer. By choosing an opportune structure, Turbo codes are able to set a fixed overhead, *i.e.*, $\epsilon > 0$, so that only $(1 + \epsilon)k$ symbols are necessary to correctly reconstruct the original dataword, while the encoding and decoding operation exhibit a complexity equal to $O(nln(1/\epsilon))$. Therefore, Turbo codes allows achieving the fast coding performance of LPDC while presenting a good correction power, closer to the one of RS codes.

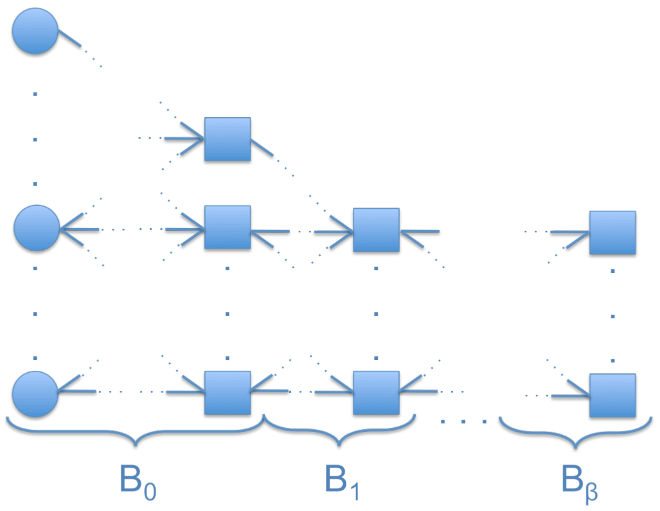


Figure A.4: Cascaded Graph Sequence that characterize the construction of Turbo Codes

A.4 Digital Fountain (DF) Codes

The main drawback of all the coding techniques presented in the previous sections is that the user has to determine the number of encoding packets, or equivalently the rate of the code ahead of time of using them. Since the message losses are unpredictable, it is tough to define the optimal rate to provide a reliable data transmission by tolerating any possible loss pattern. So, there is a not negligible probability that the decoding may fail since the received has been reached by insufficient number of packets to correctly reconstruct the original message. To eliminate the need of retransmissions to overcome the decoding failure, the most suitable approach is to implement a so-called **Digital Fountain** [99]: injecting a stream of distinct encoded packets into the network, from which the receiver can obtain the original message despite any possible loss pattern. To implement such an approach, it's needed to have coding techniques that exhibit the rateless property, *i.e.*, a potentially limitless sequence of encoding symbols can be generated from the original message by the channel encoder. Luby has shown in [99] that neither RS codes neither Turbo Codes allows to make a good approximation of a rateless code. In the first case, among the several significant limitations the most severe one is represented by the issue of the fixed alphabet[6] that forces the stretch factor of the code, *i.e.*, the redundancy applied after the encoding operation, to a maximum value assignable to n. In the second case, even if from a theoretical point of view it is possible to use a very large number of encoding packets, this is not viable in practice since the running time and memory consumption, which linearly depends on n,

[6]RS code usually follow the NASA standard to fix n to a value equal to 256 (*i.e.*, RS(255, 233, 33), so each symbol equal to one byte) since large values imply too slowly coding operations.

become too huge to be tolerated.

A.4.1 Luby Transform (LT) Codes

Luby Transform [192] (LT) codes were proposed by Luby in 2002 as a better practical realization of the ideal digital fountain. The encoding process has a very simple description: (i) a degree d is randomly chosen according to a predetermined distribution $\Omega(d)$, and (ii) d distinct symbols of the data word are uniformly selected at random, and the symbol of the codeword is obtaining by performing an exclusive-or of the selected symbols. Such codes exhibit a representation by means of a Tanner graph, but they differ from the LDPC codes since the graph is implicit rather than explicit: each encoding symbols tracks its own neighbors without a global view of the graph by carrying on additional information. The condition to decode successfully is that sender and receiver have to agree on the pseudorandom generator used to set the degree d. After such agreement phase, the decoding phase is virtually same as Turbo codes. Considering the quality of such codes, we can state that the coding operations exhibit a complexity equals to $O(n \cdot ln(n))$ where k is the number of symbols in the codeword, while the receiver can obtain the original message when receiver any $k + o(k)$ encoding symbols.

A.4.2 Raptor Codes

One drawback of the LT codes (apart from not being systematic) is that they cannot be encoded with constant costs if the number of the collected output symbols is close to the number of input symbols. **Raptor** codes [193] has been designed to overcome such disadvantage, so it aims at performing encoding and decoding with a constant cost. The

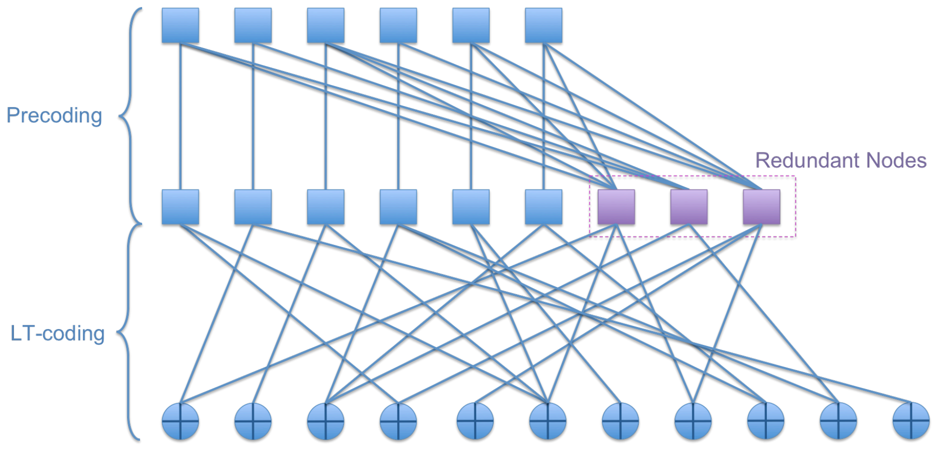


Figure A.5: Overview of a raptor code

driving idea of the raptor codes is to perform the encoding in two steps, as illustrated in Figure A.5: (i) in the first step, called *precoding*, the dataword M is encoded with a fixed erasure code such as a Tornado, and (ii) in the second step, the encoded message produced by the precoding, namely M' is treated as a dataword and a LT code is applied in order to obtain the codeword, namely M'' to exchange with the destination. When the codeword M'' reached the destination by passing through an erasure channel, it is not needed to recover all the packets of M', but only is constant fraction. In LT codes, the average degree must be at least $\Omega(ln(k))$ in order to have a decoding time reduced to $O(k)$, however, by using a raptor code this bound on the average degree is not necessary. Such codes are characterized by a decoding time equal to $O(n \cdot ln(1/\epsilon))$ and a successful decoding achieved when $(1+\epsilon) \cdot k$ symbols received where ϵ is a not negative constant. Such codes do not only have a good asymptotic performance, but exhibit a systematic version in order to allow better performance.

Table A.1: Overview of the features of the different coding approaches

Coding solution	Performance Complexity	Correcting Capacity	Rate
RD codes	High $O(n^2)$	High k	Low limited $n < 2^L$
Turbo codes	Low $O(n \cdot ln(1/\epsilon))$	Medium $(1+\epsilon) \cdot k$	Medium unlimited but not rateless
LT codes	Medium $O(n \cdot ln(n))$	Low $k + o(k)$	High rateless
Raptor codes	Low $O(n \cdot ln(1/\epsilon))$	Medium $(1+\epsilon) \cdot k$	High rateless

A.5 Discussion

The several coding techniques discusssed in the previous sections can be compared according to three quality measures: (*i*) Computational Complexity, *i.e.*, the inherent difficulty to perform the encoding and decoding operations, (*ii*) Correcting Capacity, *i.e.*, the number of needed symbols to correctly reconstruct a dataword, and (*iii*) Rate, *i.e.*, characterization of the applicable redundancy. The comparison of the coding techniques according to these three measures is presented in Table A.1. With respect to computational complexity, RS codes exhibit the worst value since its complexity is quadratic to the number of symbols in the codeword, while the others exhibit a linear trend: Specifically, Turbo and Raptor codes have a complexity equal to $O(n \cdot ln(1/\epsilon))$ with ϵ is a non-negative constant that can be made small at will with an appropriate design so that $O(n \cdot ln(1/\epsilon)) \approx O(n \cdot ln(1/\epsilon))$, while LT codes has a complexity equal to $O(n \cdot ln(n))$ which is slightly higher than a linear trend. Therefore, as shown in Figure A.6, RS codes is characterized by the highest complexity, Turbo and Erasure codes by the lowest, while LT code presents a complexity that is in

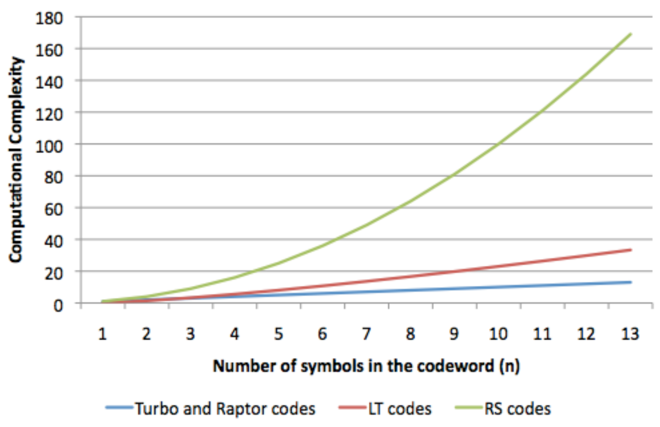


Figure A.6: Comparison among the computational complexity trends

the middle of the previous ones (even if closer to the linear trend). With respect to the correcting capacity, RS codes guarantee that the dataword is successfully reconstructed if the destination is reached by encoded symbols in number not fewer than the number of symbols in the dataword. Such capacity is the optimal one, but other codes do not provide a better capacity since they asyntotically try to converge to the capacity of RS codes. Specifically, the additional (to the number of symbols in the dataword) symbols needed by LT code is equal to $o(k)$ while for Turbo and Erasure codes it is $\epsilon \cdot k$. Since ϵ can be made small at will, we assume that the former codes exhibit a capacity closer to the one of RS codes than the LT codes. With respect to the rate, RS codes are characterized by a condition that limits the redundandy degree that can be applied by the source, in fact, $n < 2^L$ where L is the size of the Galois Field. Turbo codes do not have such limitations, but do not provide a good approximation of the rateless codes, while LT and Raptor codes are practical realization of rateless codes, so have the best rate. These considerations are summarized in Figure A.7, which clearly express the superiority of Raptor codes over the

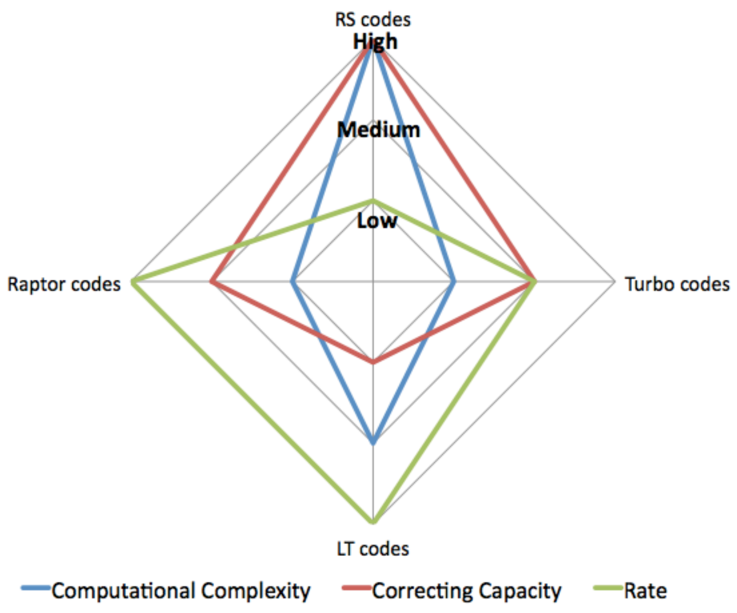


Figure A.7: Comparison among the overviewed coding techniques

other ones since they offer the best tradeoff among complexity, capacity and rate.

Appendix B

Simulation Environment

This appendix contains further details on the simulation environment adopted to assess the quality of the strategies proposed in this PhD dissertation. Specifically, models are built using the general-purpose simulation environment called Omnet++, while the networking devices are emulated using the INET framework. On the other hand, how to interconnect routers and end-hosts is treated by using the ReaSE tool, while OverSim is adopted to model the underlaying Pastry substrate, which is used in TODAI for the construction of multicast trees.

B.1 OMNET++

Omnet++ [194] is a object-oriented modular discrete event simulation environment that has been widely adopted in several application domains, *e.g.*, modeling wired and wireless communication networks, multiprocessor and parallel architectures and/or evaluating performance aspects of complex software systems. The driving idea of its creators is to to fill the gap between open-source, academic simulation software such as NS-2[1] and expensive commercial solutions such as Opnet[2] by providing an infrastructure and a formalism to write simulations [195]. In fact, Omnet++ has not been designed to simulate anything concrete, but to provide the tools for create component-oriented reusable entities, called

[1]www.isi.edu/nsnam/ns/
[2]www.opnet.com/

modules, that can be combined such as the LEGO blocks in order to made complex simulations models. Such modularity is the reason behind its growing success within the research community since new models can be realized considering off-the-shelf models provided by third parties. This section aims at providing an overview of OMNET++:

1. subsection B.1.1 illustrates the modular organization within a simulation model;
2. subsection B.1.2 discusses how distinct modules are interconnected and how such interconnections are expressed;
3. subsection B.1.3 presents the steps required to obtain an executable model;
4. subsection B.1.4 talks about the simulation kernel that is in charge of executing the models implemented by the users.

B.1.1 Modular Architecture

As said previously, an Omnet++ model consists of *modules* written in C++, which are entities that communicate with message passing. The active modules are termed *simple modules*, which contain the algorithms expressing as C++ procedures the module behavior and can be grouped into *compound modules* and so forth as shown in Figure B.1; the number of hierarchy levels is not limited. The concept of simple and compound modules is similar to DEVS atomic and coupled models [196]. The whole model, called *network*, is itself a compound module. Messages can be sent either via *connections* that span between modules or directly to other modules. Simple modules typically send messages via *gates*, but it is also possible to send them directly to their destination modules. Gates are the input and

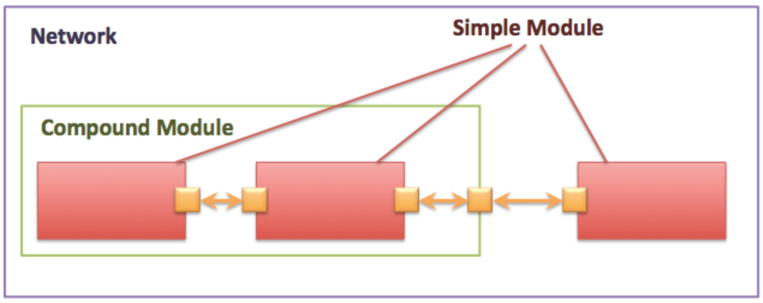


Figure B.1: Layered implementation of the simulation model

output interfaces of modules: messages, which contain arbitrarily complex data structures that are formulated in specific files with .msg extension, are sent out through output gates[3] and arrive through input gates, while an input and an output gate can be linked with a connection, which exhibits parameters such as propagation delay, data rate and bit error rate. Connections are created within a single level of module hierarchy: within a compound module, corresponding gates of two submodules, or a gate of one submodule and a gate of the compound module can be connected. Such hierarchy is not only expressed using containment relations, but also inheritance, in fact, a module can be the specialization of another distinct one. Connections spanning across hierarchy levels are not permitted, as it would hinder model reuse. Due to the hierarchical structure of the model, messages typically travel through a chain of connections, to start and arrive in simple modules. Compound modules act as 'cardboard boxes' in the model, transparently relaying messages between their inside and the outside world.

[3]An interesting feature is that the "local simulation time" of a module advances only when the module receives a message.

B.1.2 NED Language

The user defines the structure of the model in *NEtwork Description* (NED) language, *i.e.*, the NED file of a model define the modules, assemble them into compound modules and how they are interconnected through channels. The NED language has several features which let it scale well to large projects: NED files can be broken down simpler NED files that can be included in the main one. Moreover, this feature is supported by the component-based organization of the models in modules, which are inherently reusable, reduces code copying, and allows component libraries. In addition, the NED language features a Java-like package structure, to reduce the risk of name clashes between different models. Due to the hierarchical nature of the modules of a model, the NED language has an equivalent tree representation which can be serialized to XML; that is, NED files can be converted to XML and back without loss of data, including comments. This lowers the barrier for programmatic manipulation of NED files, for example extracting information, refactoring and transforming NED, generating NED from information stored in other system like SQL databases, and so on. For more information on the syntax and semantics of the NED language, refer to the Omnet++ User Manual at www.omnetpp.org/documentation.

B.1.3 Engineering Simulation Programs

The life cycle of a simulation model in Omnet++ is illustrated in Figure B.2, and it is composed of several distinct phases. To build an executable simulation program, you first need to translate the MSG files into C++, using the message compiler (opp_msgc). After this step, the process is the same as building any C/C++ program from source: all C++

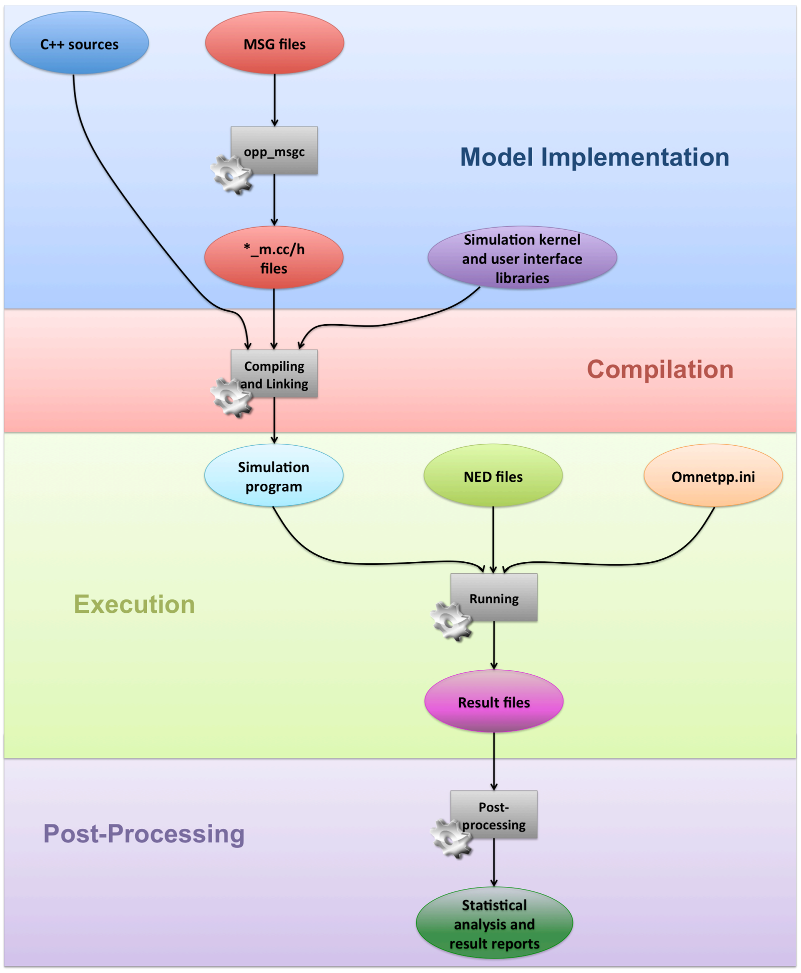


Figure B.2: Building and running simulation in Omnet++

sources need to be compiled into object files (.o files using gcc on Mac, Linux or mingw on Windows) and all object files need to be linked with the necessary libraries to get an executable or shared library. Specifically, simulation codes needs to be linked with the following libraries:

- The simulation kernel and class library, called oppsim.
- User interfaces. The common part of all user interfaces is the oppenvir library, and

the specific user interfaces are opptkenv and oppcmdenv. Moreover, codes needs also to link with oppenvir, plus opptkenv or oppcmdenv or both.

Luckily, users do not have to worry about the above details, because automatic tools like opp_makemake will take care of the hard part for the user.

By default, the output of an opp_makemake-generated makefile is a simulation executable that can be run directly. Simulations are configured in a file usually called omnetpp.ini, where values are assigned to the module parameters and the NED file that express the network module to be used for the simulation. In simple cases, this executable can be run without command-line arguments, but usually one will need to specify options to specify what ini file to use, which user interface to activate, where to load NED files from, and so on. Simulations of large scale systems, which are composed by thousands of modules, may be characterized by an overwhelming number of parameters, to reduce the size of the omnetpp.ini file, Omnet++ supports including an ini file in another, via the include keyword, so to to partition large ini files into logical units with fixed and varying parts. The user can graphically follows the execution of the simulation by Tkenv, which is a portable graphical windowing user interface. It supports interactive execution of the simulation, tracing and debugging by allowing the user to get a detailed picture of the state of simulation at any point of execution and to follow what happens inside the network. The execution of simulations generate output vectors, which contains time series data, *i.e.*, values with timestamps, and can be used to record anything that is useful to get a full picture of what happened in the model during the simulation run, *i.e.*, end-to-end delays

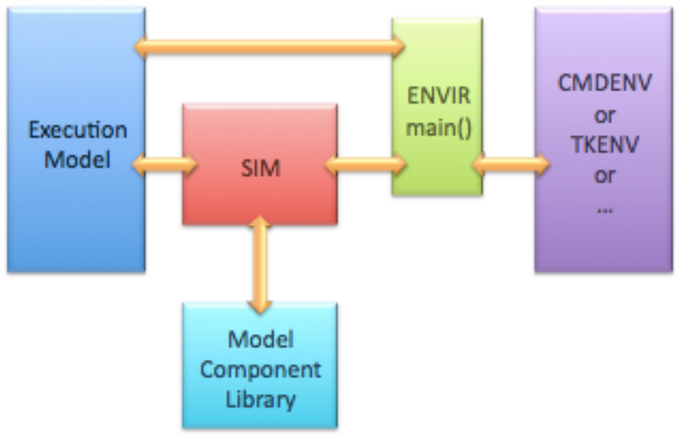


Figure B.3: Building and running simulation in Omnet++

or round trip times of packets, queue lengths, queueing times, link utilization, the number of dropped packets. Such files can be automatically processed by an Analysis Tool, namely the scavetool program, for statistical analysis and visualization of simulation results.

B.1.4 Simulation Kernel

Omnet++ has a modular architecture, which is shown in the Figure B.3 and is composed of the following components:

- *Sim* is the simulation kernel and class library that is linked to the simulation program written by the user;
- *Envir* is another library which contains all code that is common to all user interfaces, such as main(): it provides services like ini file handling for specific user interface implementations and presents itself towards Sim and the executing model via the ev facade object, hiding all other user interface internals;
- *Cmdenv* and *Tkenv* are specific user interface implementations, and a simulation is

linked with Cmdenv, Tkenv, or both;

- The *Model Component Library* consists of simple module definitions and their C++ implementations, compound module types, channels, networks, message types and in general everything that belongs to models and has been linked into the simulation program made by the user;
- The *Executing Model* is the model that has been set up for simulation. It contains objects (modules, channels, etc.) that are all instances of components in the model component library.

In figure, the arrows show how components interact with each other:

- Executing Model <==> Sim - The simulation kernel manages the future events and invokes modules in the executing model as events occur. The modules of the executing model are stored in the main object of Sim, *i.e.*, class cSimulation. In turn, the executing model calls functions in the simulation kernel and uses classes in the Sim library.
- Sim <==> Model Component Library - The simulation kernel instantiates simple modules and other components when the simulation model is set up at the beginning of the simulation run. It also refers to the component library when dynamic module creation is used. The machinery for registering and looking up components in the model component library is implemented as part of Sim.
- Executing Model <==> Envir - The ev object, logically part of Envir, is the facade

of the user interface towards the executing model. The model uses ev to write debug logs (ev<<, ev.printf()).

- Sim <==> Envir - Envir is in full command of what happens in the simulation program. Envir contains the main() function where execution begins. Envir determines which models should be set up for simulation, and instructs Sim to do so. Envir contains the main simulation loop (determine-next-event, execute-event sequence) and invokes the simulation kernel for the necessary functionality (event scheduling and event execution are implemented in Sim). Envir catches and handles errors and exceptions that occur in the simulation kernel or in the library classes during execution. Envir presents a single facade object (ev) that represents the environment (user interface) toward Sim – no Envir internals are visible to Sim or the executing model. During simulation model setup, Envir supplies parameter values for Sim when Sim asks for them. Sim writes output vectors via Envir, so one can redefine the output vector storing mechanism by changing Envir. Sim and its classes use Envir to print debug information.

- Envir <==> Tkenv/Cmdenv - Tkenv and Cmdenv are concrete user interface implementations. When a simulation program is started, the main() function (which is part of Envir) determines the appropriate user interface class, creates an instance and runs it by invoking its run() method. Sim's or the model's calls on the ev object are delegated to the user interface.

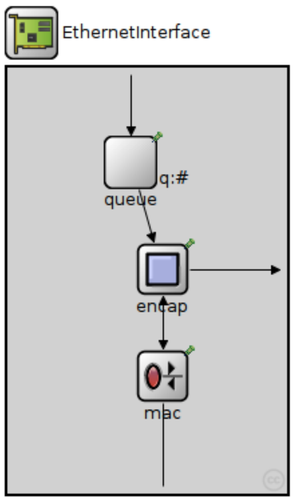


Figure B.4: Overview of the EthernetInterface provided by the INET framework

B.2 INET Framework

The **INET Framework**[4] is an open-source communication networks simulation package for the OMNeT++/OMNEST simulation environment that contains models for several Internet protocols, *e.g.*, UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, IEEE 802.11, MPLS, OSPF, and several other protocols. These modules can be freely combined to form hosts and other network devices with the NED language (no C++ code and no recompilation required). Various pre-assembled host, router, switch, access point, etc. models can be found in the Nodes/ subdirectory (*e.g.*, StandardHost, Router), but you can also create your own ones for tailored to your particular simulation scenarios. Network interfaces (*e.g.*, Ethernet, 802.11, etc) are usually compound modules (*i.e.*, assembled from simple modules) themselves, and are being composed of a queue, a MAC, and possibly other simple modules, a practical example, namely EthernetInterface, is illustrated in Figure B.4. Not

[4]inet.omnetpp.org/

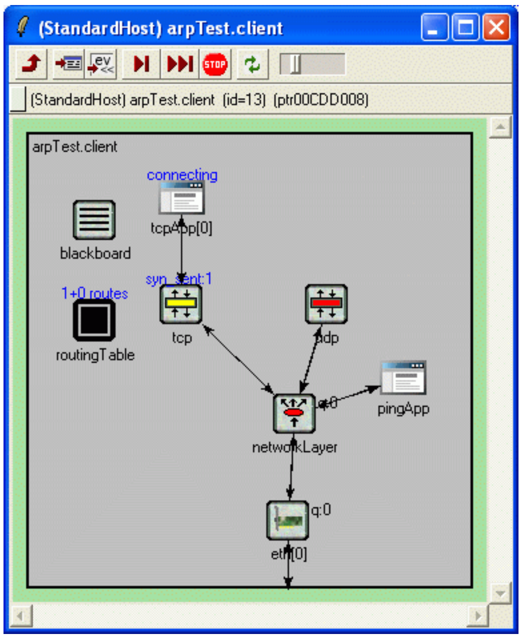


Figure B.5: Client module of the ARP protocol

all modules implement protocols though, in fact, there are modules which hold data (*e.g.*, RoutingTable), facilitate communication of modules (*e.g.*, NotificationBoard), perform autoconfiguration of a network (*e.g.*, FlatNetworkConfigurator), move a mobile node around (*e.g.*, ConstSpeedMobility), and perform housekeeping associated with radio channels in wireless simulations (*e.g.*, ChannelControl). Protocol headers and packet formats are described in message definition files (msg files), which are translated into C++ classes by OMNeT++'s opp_msgc tool, which are subclasses from OMNeT++'s cMessage class.

B.3 Rease Topology Generator

ReaSE[5] is a tool developed for creation of as realistic as possible environments for the discrete event simulator OMNeT++ in combination with the INET framework. It is based

[5]www.oversim.org/

on current state of the art solutions and considers multiple aspects: (*i*) topology generation - on AS-level as well as on router-level - as presented in subsection B.3.1, (*ii*) traffic patterns, as described in subsection B.3.2, and attack traffic, as discusses in subsection B.3.3. In order to achieve these aspects ReaSE consists of various components [160]:

1. An extension of the INET framework that enables hierarchical addressing and routing as well as generation of self-similar background traffic, furthermore, additional entities are integrated into INET, *e.g.*, a DDoSZombie or attack detection systems;

2. *ReaSEGUI*: A front-end graphical user interface that allows for creation of NED topology files that emulate realistic connectivity within LANs or WANs;

3. *TGM*: A C++ implementation that generates the basic hierarchical topologies

4. Some additional perl scripts that are used by ReaSEGUI in order to extend the basic topology by additional entities.

B.3.1 Topology Generation

The process of AS topology generation is based applying the PFP model described in subsection4.2.1.1 and takes an XML-based configuration file as input. Based on given parameters, *e.g.*, the number of ASes to generate or values for some parameters of the PFP model, ReaSE creates a single NED file that defines the required number of Autonomous Systems and their interconnections according to the topology generated by the PFP model. Each transit and stub AS is included into the compound module Internet, which is actually the root of the hierarchy that formulate the Internet topology, and additionally, each AS

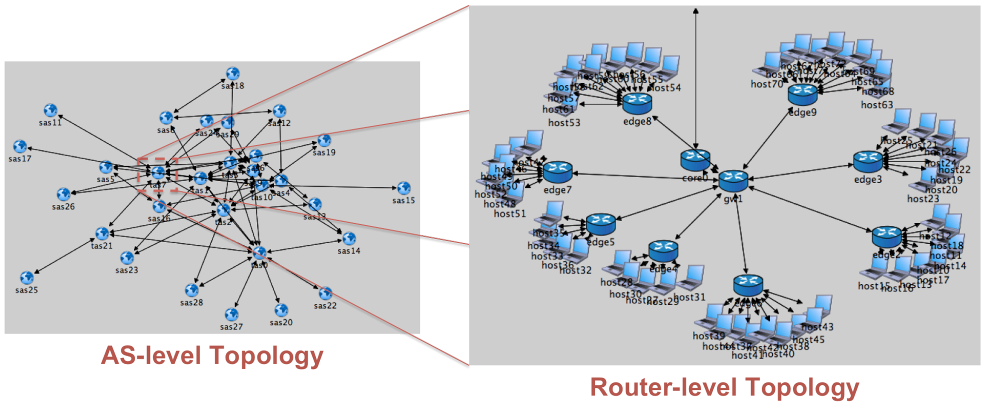


Figure B.6: Example of AS-level and router-level topology

may contain its own router level topology. The channels between different ASes are assigned a constant bandwidth that may differ between transit/transit, stub/transit, and stub/stub interconnections. This means that the delay with ReaSE currently does not depend on a node's geographic position.

Based on the NED file created during AS topology generation, in a second step each AS is filled with independently created HOT router level topologies of varying sizes, as illustrated in subsection 4.2.1.2. Therefore, each node of the router level topology is realized either by the module Router or StandardHost of the INET framework. The differentiation between different node types, *e.g.*, core and gateway routers, is achieved by assigning different link bandwidths, *e.g.*, to core/core or core/gateway channels.

B.3.2 Traffic Generation

In order to support generation of realistic traffic patterns, ReaSE extended the INET framework of OMNeT++ by additional modules: the HierarchicalNetworkConfigurator, which creates static routes within each AS as well as between different ASes based on shortest paths between source and destination, TrafficProfileManager, which reads all available traffic profiles from a XML-based configuration file, and a global as well as local ConnectionManagers per AS, which register IP address and role of every server module. The assignment of a traffic profile to a given pair of server and client is performed by the TrafficProfileManager as follows:

- Random selection of a traffic profile based on the given selection probabilities.
- Choice if the traffic flow takes place between the client and a server within the client's AS or beyond the AS boundaries. This decision is taken based on the profile's WAN selection probability.
- Notification of the selected traffic profile to the AS-specific ConnectionManager.

Multiple Omnet++ simulations presented in [160] and based on a topology with 90.000 hosts within 30 Autonomous Systems using 8 traffic profiles proved that the traffic generated with ReaSE really shows self-similar behavior, which is the requirement to satisfy in order to create realistic traffic patterns.

B.3.3 Attack Traffic

Simulation of *Distributed Denial-of-Service* (DDoS) attacks has been enabled in ReaSE by integrating in the context of the Omnet++ simulation environment a real tool for conducting such attacks: the *Tribe Flood Network* [197]. In case of worm propagations, ReaSE has implemented two different alternatives: the first one is based on UDP, the latter on TCP, both of them based on a rather simple probing mechanism that was, *e.g.*, used with Code Red I [198].

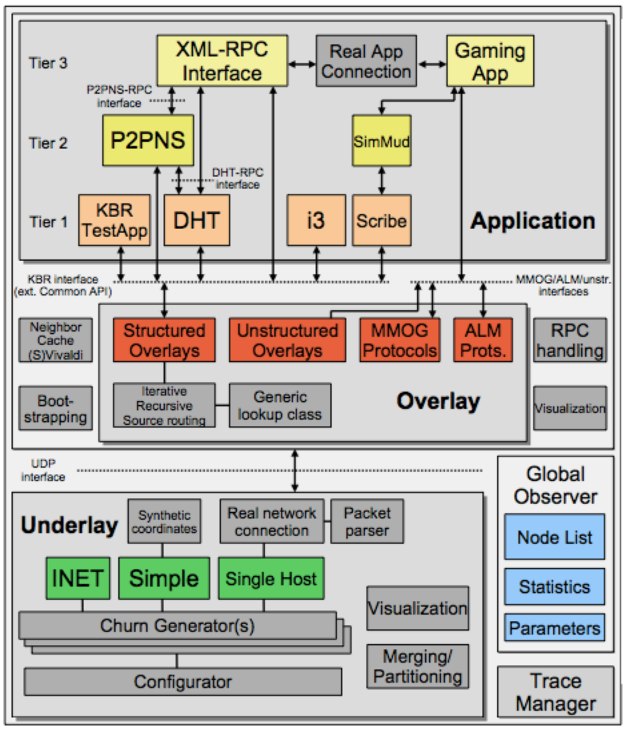


Figure B.7: Architecture of Oversim

B.4 Oversim

OverSim[6] is an open-source overlay network simulation framework for the OMNeT++/OMNEST simulation environment. It is characterized by the modular architecture shown in Figure B.7 that allows the modeling of all components of a P2P network and is composed of three distinct tiers:

- *Underlay Level* provides different models for underlay abstraction which differ in complexity and accuracy and are fully transparent to the upper overlay layer: (*i*) the

[6]www.oversim.org/

Simple Underlay is the default underlay model for OverSim, and combines a low computational overhead with high accuracy; (*ii*) INET is based on the INET framework where the IP stack is completely modeled and even routers can be part of the simulated overlay; (*iii*) the SingleHost Underlay provides real network support for OverSim by acting as a middleware to support deploying overlay protocols developed for OverSim on real networks. In Sect. IV, this feature will be discussed in greater detail.

- *Overlay Level* provides several functions that many overlay protocol implementations (*e.g.*, Chord, Kademlia, Pastry) have in common: (*i*) overlay message handling (*e.g.*, RPCs), (*ii*) generic lookup with support for different routing modes, (*iii*) node failure discovery and routing table recovery, (*iv*) common API interface support, (*v*) bootstrapping support and (*vi*) proximity awareness (*e.g.*, Vivaldi, GNP). All of these features allow for rapid overlay protocol prototyping and make protocol implementations comparable and less error-prone.

- *Application Level* consists of business applications that communicate through the underlaying overlay protocol. Additionally OverSim makes use of an XML-RPC interface to provide overlay services (*e.g.*, distributed data storage) to external applications similar to the interface provided by the OpenDHT service.

- *Modeling of churn* provides several models for generating churn supporting different distribution functions (*e.g.*, Weibull, Pareto or Exponential). Alternatively, a scenario or trace file containing join and leave events can be used to model churn behavior, which allows to easily generate complex scenarios with heterogeneous node behavior.

Bibliography

[1] D. Bailey and E. Wright. *Practical SCADA for industries.* Newnes, 2003.

[2] S. Bologna, C. Balducelli, G. Dipoppa, and G. Vicoli. Dependability and Survivability of Large Complex Critical Infrastructures. *Computer Safety, Reliability, and Security, Lecture Notes in Computer Science*, 2788:342–353, September 2003.

[3] L. Northrop et al. *Ultra-large-scale Systems, The Software Challenge of the Future.* Software Engineering Institute, Carnegie Mellon University, http://www.sei.cmu.edu/uls/, 2006.

[4] Sun Microsystems. Java Message Service, v1.1. *SUN Specification*, 2002.

[5] Object Management Group. Data Distribution Service (DDS) for Real-Time Systems, v1.2. *OMG Document*, 2007.

[6] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, and R. Morris. Resilient Overlay Networks. *ACM Operating Systems Review (SIGOPS)*, 35(5):131–145, December 2001.

[7] N. Feamster, D.G. Andersen, H. Balakrishnan, and M.F. Kaashoek. Measuring the Effects of Internet Path Faults on Reactive Routing. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):126–137, June 2003.

[8] A. Markopoulou, F. Tobagi, and M. Karam. Loss and Delay Measurements of Internet Backbones. *Computer Communications*, 29(10):1590–1604, September 2003.

[9] F. Wang, Z.M. Mao, J. Wang, L. Gao, and R. Bush. A measurement study on the impact of routing events on end-to-end internet path performance. *Computer Communications*, 36(4):375–386, October 2006.

[10] K. M. Chandy, M. Charpentier, and A. Capponi. Towards a theory of events. *Proceedings of the Inaugural International Conference on Distributed Event-Based Systems (DEBS 07)*, pages 180–187, June 2007.

[11] I. Gupta, K. P. Birman, and R. van Renesse. Fighting Fire with Fire: Using Randomized Gossip to Combat Stochastic Scalability Limits. *Special number of Quality and Reliability of Computer Network Systems, Journal of Quality and Reliability Engineering International*, 18(3):165–184, May/June 2002.

[12] P.Th. Eugster, P.A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many Faces of Publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, June 2003.

[13] Thales. TACTICOS - A data-centric Combat Management System. *OMG Real-time Workshop*, 2005.

[14] M.A. Jaeger, G. Muhl, M. Werner, H. Parzyjegla, and H.-U. Heiss. Algorithms for Reconfiguring Self-Stabilizing Publish/Subscribe Systems. *Autonomous Systems - Self-Organization, Management, and Control, Springer*, pages 135–147, September 2008.

[15] G. Cugola and G.P. Picco. REDS: a reconfigurable dispatching system. *Proceedings of the 6th international workshop on Software Engineering and Middleware (SEM 06)*, pages 9–16, November 2006.

[16] P. Costa, M. Migliavacca, G.P. Picco, and G. Cugola. Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation. *Proceeding of the 24th IEEE International Conference on Distributed Computing Systems (ICDCS 04)*, pages 552–561, March 2000.

[17] R. Chand and P. Felber. XNET: a Reliable Content-based Publish/Subscribe System. *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS 04)*, pages 264– 273, October 2004.

[18] L. Rizzo and L. Vicisano. RMDP: an FEC-based reliable multicast protocol for wireless environments. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R 98)*, 2(2):23–31, April 1998.

[19] V. Venkataraman, K. Yoshida, and P. Francis. Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast. *Proceedings of the 2006 14th IEEE International Conference on Network Protocols (ICNP 06)*, pages 2–11, November 2006.

[20] J. Han and F. Jahanian. Impact of Path Diversity on Multi-homed and Overlay Networks. *Proceedings of the International Conference on Dependable Systems and Networks (DSN 04)*, pages 29–40, July 2004.

[21] C. Esposito, D. Cotroneo, and A. Gokhale. Reliable Publish/Subscribe Middleware for Time-sensitive Internet-scale Applications. *Proceedings of the 3rd ACM International Conference on Distributed Event-Based Systems (DEBS 09)*, July 2009.

[22] R. Ahlswede, S.-Y.R. Ning Cai Li, and R.W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory (TIT)*, 46(4):298–313, July 2000.

[23] P. Marwedel. *Embedded System Design.* Springer, 2006.

[24] I. Koren and C. Mani Krishna. *Fault-Tolerant Systems.* Morgan Kaufmann, 2007.

[25] D. C. Schmidt. Middleware for Real-time and Embedded Systems. *Communications of the ACM*, 45(6):43–48, June 2002.

[26] T. C. Yang. Networked Control System: a Brief Survey. *IEEE Proceedings on Control Theory and Applications*, 153(1):403–412, July 2006.

[27] E. A. Lee. What's Ahead for Embedded Software? *IEEE Computer*, 33(9):18–26, September 2000.

[28] G. J. Pottie and W. J. Kaiser. *Principles of Embedded Networked Systems Design.* Cambridge University Press, 2008.

[29] F. F. Wu, K. Moslehi, and A. Bose. Power System Control Centers: Past, Present and Future. *Proceedings of the IEEE*, 93(11):1890–1908, November 2005.

[30] F.-L. Lian, J. Moyne, and D. Tilbury. Network Design Considerations for Distributed Control Systems. *IEEE Transactions on Control Systems Technology*, 10(2):297–307, March 2002.

[31] Terna. Dati Statistici sull'energia elettrica in Italia, Dati Generali. *http://www.terna.it/LinkClick.aspx?fileticket=OnkycVUaLqs%3d&tabid=418&mid=2501*, 2007.

[32] EuroControl. EATMS Operational Concept Document, ver. 1.1. *EuroControl ODT Library*, 1998.

[33] EuroCONTROL. The ATM Deployment Sequence. *SESAR Project Milestone Deliverable D4*, February 2008.

[34] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and Christophe Diot. Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Transactions on Networking (TON)*, 16(4):749–762, August 2008.

[35] D.C. Sharp. Reducing Avionics Software Cost Through Component Based Product Line Development. *Proceedings of the 10th Annual Software Technology Conference*, 1998.

[36] Q. H. Mahmoud. *Middleware for Communications.* Wiley, 2004.

[37] A. Birrell and B. Nelson. Implementing Remote Procedure Calls. *ACM Transactions on Computer Systems*, 2(1):39–59, February 1984.

[38] M. Stal. Web Services: Beyond Component-based Computing. *Communications of the ACM*, 45(10):71–76, October 2002.

[39] G. Muhl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems.* Springer, 2006.

[40] T. Faison. *Event-Based Programming - Taking Events to the Limit.* Apress, 2006.

[41] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns.* Addison-Wesley, 1995.

[42] D. Riehle. The Event Notification Pattern - Integrating Implicit Invocation with Object-orientation. *Theory and Practice of Object Systems*, 2(1):43–52, 1996.

[43] D. Garlan ad D. Notkin. Formalizing Design Spaces: Implicit Invocation Mechanisms. *Lecture Notes in Computer Science: VDM'91 Formal Software Development Methods*, 551/1991:31–44, 1991.

[44] D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects.* Wiley, 2000.

[45] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns.* Wiley, 1996.

[46] Object Management Group (OMG). CORBA Event Service Specification, ver. 1.0. *OMG Document formal/2000-06-15*, 2000.

[47] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen. The Information Bus: an Architecture for Extensible Distributed Systems. *ACM SIGOPS Operating Systems Review*, 27(5):58–68, December 1993.

[48] D. Rosenblum and A. Wolf. A Design Framework for Internet-scale Event Observation and Notification. *ACM SIGSOFT Software Engineering Notes*, 22(6):344–360, November 1997.

[49] P.Th. Eugster. Type-based publish/subscribe: Concepts and experiences. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29(1):1–50, January 2007.

[50] Object Management Group (OMG). CORBA Notification Service Specification, ver. 1.1. *OMG Document formal/2004-10-11*, 2004.

[51] H.-A. Jacobsen, A. Cheung, G. Li, B. Maniymaran, V. Muthusamy, and R. S. Kazemzadeh. The PADRES Publish/Subscribe System. *Book chapter of Handbook of Research on Advanced Distributed Event-Based Systems, Publish/Subscribe and Message Filtering Technologies, a book edited by A. Hinze and A. Buchmann and scheduled to be published by IGI Global*, 2009.

[52] P. Veríssimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2004.

[53] T. Schlossnagle. *Scalable Internet Architectures*. Sams, 2006.

[54] R. Meier and V. Cahill. Taxonomy of Distributed Event-Based Programming Systems. *The Computer Journal*, 48(4):602–626, June 2005.

[55] S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems (TCS)*, 8(2):85–100, May 1990.

[56] J. F. Buford, H. Yu, and E. K. Lua. *P2P Networking and Applications*. Morgan Kaufmann, 2008.

[57] C. O'Ryan, D. L. Levine, D. C. Schmidt, and J. R. Noseworthy. Applying a Scalable CORBA Event Service to Large-Scale Distributed Interactive Simulations. *Proceedings of the Fifth International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS)*, pages 67–74, 1999.

[58] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems (TOCS)*, 19(3):332–383, August 2001.

[59] G. V. Chockler, I. Keidar, and R. Vitenberg. Group Communication Specifications: A Comprehensive Study. *ACM Computing Surveys (CSUR)*, 33(4):427–469, December 2001.

[60] T. K. Moon. *Error Correction Coding - Mathematical Methods and Algorithms*. Wiley, 2006.

[61] J. Zhang, L. Liu, L. Ramaswamy, and C. Pu. PeerCast: Churn-resilient End System Multicast on Heterogeneous Overlay Networks. *Journal of Network and Computer Applications*, 31(4):821–850, November 2008.

[62] B. N. Levine and J. J. Garcia-Luna-Aceves. A comparison of reliable multicast protocols. *Multimedia Systems*, 6(5):334–348, September 1998.

[63] X. Jin, W. P. Ken Yiu, and S. H. Gary Chan. Loss Recovery in Application-Layer Multicast. *IEEE MultiMedia*, 15(1):18–27, January 2008.

[64] A. Popescu, D. Constantinescu, D. Erman, and D. Ilie. A survey of reliable multicast communication. *Proceedings of the 3rd EuroNGI Conference on Next Generation Internet Networks*, pages 111–118, May 2007.

[65] K. Obraczka. Multicast transport protocols: A survey and taxonomy. *IEEE Communications Magazine*, 36(1):94–102, January 1998.

[66] C. K. Yeo, B. S. Lee, and M. H. Er. A survey of application level multicast techniques. *Computer Communications*, 27(15):1547–1568, September 2004.

[67] M. Hosseini, D. Tanvir, S. Shimohammadi, and N. D. Georganas. A survey of application-layer multicast protocols. *IEEE Communications Surveys & Tutorials*, 9(3):58–74, Third Quarter 2007.

[68] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A Case for End System Multicast. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1456–1471, October 2002.

[69] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. The PIM Architecture for Wide-Area Multicast Routing. *IEEE/ACM Transactions on Networking (TON)*, 4(2):784–803, December 1997.

[70] C. Diot, B.N. Levine, B. Lyles, H. Kassan, and D. Balendiefen. Deployment numbers for the IP Multicast services and architecture. *IEEE Networks - Special number Multicasting*, 14(1):78–88, 2000.

[71] M. Balakrishnan, S. Pleisch, and K. P. Birman. Slingshot: Time-critical Multicast for Clustered Applications. *Proceedings of the 4th IEEE International Symposium on Network Computing and Applications (NCA 05)*, pages 205–214, July 2005.

[72] Y. Chawathe, S. McCanne, and E. A. Brewer. RMX: Reliable Multicast for Heterogeneous Networks. *Proceedings of the 19th Conference on Computer Communications (INFOCOM 00)*, pages 795–804, March 2000.

[73] F. Baccelli, A. Chaintreau, Z. Liu, A. Riabov, and S. Sahu. Scalability of Reliable Group Communication Using Overlays. *Proceedings of the 23th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 04)*, pages 419–430, March 2004.

[74] M.S. Borella, D. Swider, S. Uludag, and G.B. Brewster. Internet Packet Loss: Measurement and Implications for End-to-End QoS. *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW 98)*, pages 3–13, 1998.

[75] D. Loguinov and H. Radha. Measurement Study of Low-Bitrate Internet Video Streaming. *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement (IMW 01)*, pages 281–293, 2001.

[76] J. C. Lin and S. Paul. RMTP: A Reliable Multicast Transport Protocol. *Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer Societies (INFOCOM 96)*, pages 1414–1424, March 1996.

[77] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Lightweight Sessions and Application-Level Framing. *IEEE/ACM Transactions on Networking (TON)*, 5(6):784–803, December 1997.

[78] D. Towsley, J. Kurose, and S. Pingali. A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. *IEEE Journal on Selected Areas in Communications (JSAC)*, 15(3):398–406, April 1997.

[79] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. *ACM SIGCOMM Computer Communication Review (CCR)*, 25(4):328–341, October 1995.

[80] M. S. Lacher, J. Nonnenmacher, and E. W. Biersack. Performance Comparison of Centralized versus Distributed Error Recovery for Reliable Multicast. *IEEE/ACM Transactions on Networking (TON)*, 8(2):224–238, April 2000.

[81] M. Balakrishnan, K. P. Birman, A. Phanishayee, and S. Pleisch. Ricochet: Lateral Error Correction for Time-Critical Multicast. *Proceedings of the 4th USENIX Symposium on Networked System Design & Implementation (NSDI 07)*, pages 73–86, April 2007.

[82] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal Multicast. *ACM Transactions on Computer Systems (TCS)*, 17(2):41–88, May 1999.

[83] Q. Sun and D. C. Sturman. A Gossip-based Reliable Multicast for Large-scale High-throught Applications. *Proceedings of the 30th International Conference on Dependable Systems and Networks (formely FTCS-30 and DCCA-8, now DSN 00)*, pages 347–458, June 2000.

[84] J. Nonnenmacher, E. W. Biersack, and D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Trasmission. *IEEE/ACM Transactions on Networking (TON)*, 6(4):349–361, August 1998.

[85] C. Huitema. The Case for Packet Level FEC. *Proceeding of the IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN 96)*, pages 110–120, October 1996.

[86] N. Magharei, R. Rejaie, and Y. Guo. Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches. *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 07)*, pages 1424–1432, May 2007.

[87] A.-M. Kermarrec, L-Massoulié, and A. J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 14(2):1–11, February 2003.

[88] F. Wang, Y. Xiong, and J. Liu. mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast. *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS 07)*, pages 49–57, June 2007.

[89] P. Radoslavov, C. Papadopoulos, R. Govindan, and D. Estrin. A Comparison of Application-Level and Router-Assisted Hierarchical Schemes for Reliable Multicast. *IEEE/ACM Transactions on Networking (TON)*, 7(3):375–386, June 1999.

[90] X. R. Xu, A. C. Myers, H. Zhang, and R. Yavatkar. Resilient Multicast Support for Continuous-Media Applications. *Proceedings of the IEEE 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 97)*, pages 183–194, May 1997.

[91] W.-P. K. Yiu, K.-F. S. Wong, S.-H. G. Chan, W.-C. Wong, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang. Lateral Error Correction for Media Streaming in Application-Level Multicast. *IEEE/ACM Transactions on Multimedia (T-MM)*, 8(2):219–232, April 2006.

[92] G. Tan, S. A. Jarvis, and D. P. Spooner. Improving the Fault Resilience of Overlay Multicast for Media Streaming. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 18(6):721–734, June 2007.

[93] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. Epidemic Information Dissemination in Distributed Systems. *IEEE Computer*, 37(5):60–67, May 2004.

[94] Alan Demers et al. Epidemic Algorithms for Replicated Database Maintenance. *ACM SIGOPS Operating Systems Review (SIGOPS)*, 22(1):8–32, June 1998.

[95] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3):8–36, August 2007.

[96] J. Leitao, J. Pereira, and L. Rodrigues. Epidemic Broadcast Trees. *Proceedings of the 26rd IEEE International Symposium on Reliable Distributed Systems (SRDS 07)*, pages 301–310, October 2007.

[97] I. Gupta, A.-M. Kermarrec, and A. J. Ganesh. Efficient and Adaptive Epidemic-Style Protocols for Reliable and Scalable Multicast. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 17(7):593–605, July 2006.

[98] K. Hopkinson, K. Jenkins, K. Birman, J. Thorp, G. Toussaint, and M.Parashar. Adaptive Gravitational Gossip: A Gossip-Based Communication Protocol with User-Selectable Rates. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 20(12):1830–1843, December 2009.

[99] J. W. Byers, M. Luby, and M. Mitzenmacher. A Digital Fountain Approach to Asynchronous Reliable Multicast. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1528–1540, October 2002.

[100] M. Ghaderi, D. Towsley, and J. Kurose. Reliability Gain of Network Coding in Lossy Wireless Networks. *Proceedings of the 27th Conference on Computer Communications (INFOCOM 08)*, pages 2171–2179, April 2008.

[101] M. Wu, S. S. Karande, and H. Radha. Network-embedded FEC for Optimum Throughput of Multicast Packet Video. *Journal on Signal Processing: Image Communication*, 20(8):728–742, September 2005.

[102] J. Han, D. Watson, and F. Jahanian. An Experimental Study of Internet Path Diversity. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 3(4):273–288, October 2006.

[103] R. Teixiera, K. Marzullo, S. Savage, and G. M. Voelker. In Search of Path Diversity in ISP Networks. *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement (IMC 03)*, pages 313–318, October 2003.

[104] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP Topologies with Rocketfuel. *IEEE/ACM Transactions on Networking (TON)*, 12(1):2–16, February 2004.

[105] C. Tang and P.K. McKinley. Improving Multipath Reliability in Topology-Aware Overlay Networks. *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW 05)*, pages 82–88, June 2005.

[106] D.A.M. Villela and O.C.M.B. Duarte. Improving Scalability on Reliable Multicast Communication. *Computer Communications*, 24(5-6):548–562, March 2001.

[107] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. *Proceedings of the nineteenth ACM Symposium on Operating Systems Principles (SOSP 03)*, pages 298–313, October 2003.

[108] T. Nguyen and A. Zakhor. Path Diversity with Forward Error Correction (PDF) System for Packet Switched Networks. *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications (ISORC 03)*, pages 663–672, April 2003.

[109] A.E. Kamal and A. Ramamoorthy. Overlay Protection Against Link Failures Using Network Coding. *Proceedings of the 42nd Annual Conference on Information Sciences and Systems (CISS 08)*, pages 527–533, March 2008.

[110] A.C. Snoeren, K. Conley, and D.K. Gifford. Mesh-based Content Routing using XML. *ACM Operating Systems (SIGOPS)*, 35(5):160–173, December 2001.

[111] N. Magharei and R. Rejaie. PRIME: Peer-to-Peer Receiver-driven Mesh-based Streaming. *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 07)*, pages 1415–1423, May 2007.

[112] S. Birrer and F.E. Bustamante. A Comparison of Resilient Overlay Multicast Approaches. *IEEE Journal on Selected Areas in Communications (JSAC)*, 25(9):1695–1705, December 2007.

[113] Y. Hongyun, H. Ruiming, C. Jun, and C. Xuhui. A Review of Resilient Approaches to Peer-to-Peer Overlay Multicast of Media Streaming. *Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 08)*, pages 1–4, October 2008.

[114] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient Multicast Using Overlays. *IEEE/ACM Transactions on Networking (TON)*, 14(2):237–248, April 2006.

[115] S. Birrer and F.E. Bustamante. Resilient Peer-to-Peer Multicast without the Cost. *Proceedings of the 12th Annual Multimedia Computing and Networking Conference (MMCN 05)*, January 2005.

[116] J. Seibert, D. Zage, S. Fahmy, and C. Nita-Rotaru. Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective. *Proceedings of the the 33rd IEEE Conference on Local Computer Networks (LCN 08)*, pages 20–27, October 2008.

[117] R. Chand and P. Felber. Scalable Distribution of XML Content with XNET. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 19(4):447– 461, April 2008.

[118] J. Hoffert, D.C. Schmidt, and A. Gokhale. Evaluating Transport Protocols for Real-Time Event Stream Processing Middleware and Applications. *Proceedings of the 11th International Symposium on Distributed Objects, Middleware, and Applications (DOA'09)*, November 2009.

[119] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee. How to Model an Internetwork. *Proceedings of the 15th Annual Joint Conference of the IEEE Computer Societies (INFOCOM 96)*, 2:594–602, 1996.

[120] W. Zhao, D. Olshefski, and Henning Schulzrinne. Internet Quality of Service: An Overview. *Columbia University Research Report CUCS-003-00*, 2000.

[121] K.C. Almeroth. The Evolution of Multicast: from the MBone to Interdomain Multicast to Internet2 Deployment. *IEEE Network*, 14(1):10–20, January/February 2000.

[122] J.F. Kurose and K.W. Ross. *Computer Networking: A Top-Down Approach (5th edition).* Addison Wesley, 2009.

[123] L. Garcs-Erice, E.W. Biersack, P.A. Felber, K.W. Ross, and G. Urvoy-Keller. Hierarchical Peer-to-Peer Systems. *Parallel Processing Letters (PPL)*, 13(4):643–657, December 2003.

[124] L. Lao, J.-H. Cui, M. Gerla, and S. Cheng. A scalable overlay multicast architecture for large-scale applications. *IEEE Transactions on Parallel and Distributed Systems*, 18(4):449–459, April 2007.

[125] C.G. Plaxton, R. Rajaraman, and A.W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. *Theory of Computing Systems*, 32(3):241280, February 1999.

[126] A. Rowstrom and P. Drushel. Pastry: Scalable, Decentralized Object Localization and Routing for Large-scale Peer-to-Peer Systems. *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Lecture Notes in Computer Science*, 2218:329–351, November 2001.

[127] M. Castro, P. Drushel, A.M. Kermarec, and A. Rowstrom. Scribe: A Large-scale and Decentralized Application-level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1489–1499, January 2004.

[128] V. Paxson. End-to-End Routing Behavior in the Internet. *ACM SIGCOMM Computer Communication Review*, 36(5):41–56, October 2006.

[129] M. Dahlin, B.B.V. Chandra, L. Gao, and A. Nayate. End-to-End WAN Service Availability. *IEEE/ACM Transactions on Networking (TON)*, 11(2):300–313, April 2003.

[130] M. Dahlin, B.B.V. Chandra, L. Gao, and A. Nayate. A Measurement Study on the Impact of Routing Events on End-to-End Internet Path Performance. *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 375–386, August 2006.

[131] C. Labovitz, G.R. Malan, and F. Jahanian. Internet Routing Instability. *IEEE/ACM Transactions on Networking (TON)*, 6(5):515–528, October 1998.

[132] M. Zhang, B. Liu, and B. Zhang. Load-Balanced IP Fast Failure Recovery. *IP Operations and Management, Lecture Notes in Computer Science*, 5275:53–65, September 2008.

[133] J. Li and S. Vuong. An Efficient Clustered Architecture for P2P Networks. *Proceedings of the 18th International Conference on Advanced Information Networking and Applications (AINA 04)*, pages 278–284, March 2004.

[134] B. Beverly Yang and H. Garcia-Molina. Designing a Super-Peer Network. *Proceedings of the 19th International Conference on Data Engineering (ICDE 03)*, pages 49–60, March 2003.

[135] G. Coulouris, J. Dellimore, and T. Kindberg. *Distributed Systems - Concepts and Design (4th edition).* Addison Wesley, 2005.

[136] H. Garcia-Molina. Elections in a Distributed Computing System. *IEEE Transactions on Computers (TC)*, C-31(1):48–50, January 1982.

[137] M. Wu, S. S. Karande, and H. Radha. Network-Embedded FEC for Optimum Throughput of Multicast Packet Video. *Journal on Signal Processing: Image Communication*, 20(8):728–742, September 2005.

[138] S.L. Hakimi. Optimum Distribution of Switching Centers in a Communication Network and some Related Graph Theoretic Problems. *Operations Research*, 13(3):462–475, May-June 1965.

[139] B.C. Tansel, R.L. Francis, and T.J. Lowe. Location on Networks: A Survey - PART I: The P-Center and P- Median Problems. *Management Science*, 29(4):482–497, April 1983.

[140] Y. Shan, I. V. Bajic, S. Kalyanaraman, and J. W. Woods. Overlay Multi-hop FEC Scheme for Video Streaming. *Journal on Signal Processing: Image Communications*, 20(8):710–727, September 2005.

[141] J. Bang-Jensen, G. Gutin, and A. Yeo. When the Greedy Algorithm Fails. *Discrete Optimization*, 1(2):121–127, November 2004.

[142] M. Wu and H. Radha. Distributed Codec Placement Algorithm for Network-Embedded FEC. *Proceedings of the 40th Annual Conference on Information Sciences and Systems (CISS 06)*, pages 151–156, March 2006.

[143] Y. Shan, S. Kalyanaraman, and J. W. Woods. Distributed Fine Grain Adaptive-FEC Scheme for Scalable Video Streaming. *Proceedings of the SPIE*, 6822:1–10, January 2008.

[144] J. Han, G.R. Malan, and F. Jahanian. Fault-Tolerant Virtual Private Networks within An Autonomous System. *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS 02)*, pages 41–50, October 2002.

[145] H. Chang, S. Jamin, and W. Willinger. Internet Connectivity at the AS-level: an Optimization-driven Modeling Approach. *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, pages 33–46, August 2003.

[146] Z.M. Mao, J. Rexford, J. Wang, and R.H. Katz. Toward an Accurate AS-level Traceroute Tool. *Proceedings of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, pages 365–378, August 2003.

[147] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz. Disjoint Multipath Routing using Colored Trees. *Computer Networks*, 51(8):2163–2180, June 2007.

[148] R. Tian, Q. Zhang, Z. Xiang, Y. Xiong, X. Li, and W. Zhu. Robust and Efficient Path Diversity in Application-Layer Multicast for Video Streaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(8):961–972, August 2005.

[149] T. Nguyen. On the Disjoint Paths Problems. *Operations Research Letters*, 35(1):10–16, January 2007.

[150] S. Deb, M. Mdard, and C. Choute. Algebraic Gossip: A Network Coding Approach to Optimal Multiple Rumor Mongering. *IEEE Transactions on Information Theory*, 52(6):2486–2507, June 2006.

[151] W.-T. Tan and A. Zakhor. Video Multicast Using Layered FEC and Scalable Compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):373–386, March 2001.

[152] R. Jain. *The Art of Computer Systems Performance Analysis.* Wiley Professional Computing, 1991.

[153] F. Berman, G. Fox, and A.J.G. Hey. *Grid Computing: Making the Global Infrastructure a Reality.* Wiley Series on Communications Networking & Distributed Systems, 2003.

[154] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. *ACM SIGCOMM Computer Communication Review*, 33(1):59–64, January 2003.

[155] N. Spring, L. Peterson, A. Bavier, and V. Pai. Using PlanetLab for Network Research: Myths, Realities, and Best Practices. *ACM SIGOPS Operating Systems Review*, 40(1):17–24, January 2006.

[156] M.A. Rahman, A. Pakstas, and F.Z. Wang. Network Modelling and Simulation Tools. *Simulation Modelling Practice and Theory*, 17(6):1011–1031, July 2009.

[157] E. Weingartner, H. vom Lehn, and K. Wehrle. A Performance Comparison of Recent Network Simulators. *Proceedings of the IEEE International Conference on Communications (ICC 09)*, pages 1–5, June 2009.

[158] S. Zhou. Characterizing and Modeling the Internet Topology: the Rich-club Phenomenon and the PFP Model. *BT Technology Journal*, 24(8):108–115, July 2006.

[159] J. Hawkinson and T. Bates. Guidelines for Creation, Selection, and Registration of an Autonomous System (AS). *RFC 1930 (Best Current Practice)*, March 1996.

[160] T. Gamer and M. Scharf. Realistic Simulation Environments for IP-based Networks. *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (ICST 08)*, March 2008.

[161] B.M. Waxman. Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communications (JSAC)*, 6(9):1617–1622, December 1988.

[162] R. Albert and A.-L. Barabasi. Topology of Evolving Networks: Local Events and Universality. *Physical Review Letters*, 85(24):5234–5237, December 2000.

[163] S. Zhou. Power Law Modelling of Internet Topology. *Complex Sciences, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 5:2090–2098, June 2009.

[164] J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. *Technical Report CSE-TR-456-02, University of Michigan Technical Report*, 2002.

[165] S. Zhou and R.J. Mondragon. The Positive-Feedback Preference Model of the AS-level Internet Topology. *Proceedings of the IEEE International Conference on Communications (ICC 05)*, pages 163–167, May 2005.

[166] H. Haddadi, D. Fay, S. Uhlig, A. Moore, R. Mortier, A. Jamakovic, and M. Rio. Tuning Topology Generators Using Spectral Distributions. *Proceedings of the SPEC International Workshop on Performance Evaluation (SIPEW 08), Lecture Notes In Computer Science*, 5119:154–173, June 2008.

[167] P. Erdos and A. Renyi. On Random Graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[168] E. Zegura, K.L. Calvert, and M.J. Donahoo. A Quantitative Comparison of Graph-based Models for Internet Topology. *IEEE/ACM Transactions on Networking (TON)*, 5(6):1590–1604, December 1997.

[169] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-law Relationships of the Internet Topology. *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM 99)*, pages 251–262, August/September 1999.

[170] L. Li, D. Alderson, W. Willinger, and J. Doyle. A First-principles Approach to Understanding the Internets Router-level Topology. *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 04)*, pages 3–14, August/September 2004.

[171] A.-L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, October 1999.

[172] F. Chung and L. Lu. The Average Distance in a Random Graph with Given Expected Degrees. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 99(25):15879–15882, December 2002.

[173] W. Aiello, F. Chung, and L. Lu. A Random Graph Model for Massive Graphs. *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 00)*, pages 171–180, May 2000.

[174] S. Banerjee, T.G. Griffin, and M. Pias. The Interdomain Connectivity of PlanetLab Nodes. *Proceedings of Passive & Active Measurement (PAM 04)*, pages 73–82, April 2004.

[175] J.C. Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. *Proceedings of the Conference on Communications Architectures, Protocols and Applications (SIGCOMM 93)*, pages 289–298, 1993.

[176] W. Jiang and H. Schulzrinne. Comparison and Optimization of Packet Loss Repair Methods on VoIP Perceived Quality under Bursty Loss. *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 73–81, April 2002.

[177] D.L. Massart, J. Smeyers-Verbeke, X. Capron, and K. Schlesier. Visual Presentation of Data by Means of Box Plots. *LC-GC Europe*, 18(4):2–5, 2005.

[178] A. Bondavalli and L. Simoncini. Failures Classification with Respect to Detection. *Proceedings of the 2nd IEEE Workshop on Future Trends in Distributed Computing Systems*, pages 47–53, September/October 1990.

[179] R.K. Iyer and D. Tang. Experimental Analysis of Computer System Dependability. *Fault-tolerant Computer System Design, Prentice-Hall Advanced Computing And Telecommunications Series*, pages 282–392, 1996.

[180] Y. Huang, C.M.R. Kintala, N. Kolettis, and N.D. Fulton. Software Rejuvenation: Analysis, Module, and Applications. *Proceedings of the 25h International Symposium on Fault-Tolerant Computing (FTCS-25)*, pages 381–390, June 1995.

[181] A. Bondavalli, S. Chiaradonna, D. Cotroneo, and L. Romano. Effective Fault Treatment for Improving the Dependability of COTS and Legacy-based Applications. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 1(4):1545–5971, October-December 2003.

[182] R. Mullen. The Lognormal Distribution of Software Failure Rates: Origin and Evidence. *Proceedings of the 9th International Symposium on Software Reliability Engineering (ISSRE 98)*, pages 124–133, November 1998.

[183] D.A. Lelewer and D.S. Hirschberg. Data Compression. *ACM Computing Surveys (CSUR)*, 19(3):261296, September 1987.

[184] B. Kaliski. A Survey of Encryption Standards. *IEEE Micro*, 13(6):74–81, November 1993.

[185] E. Biglieri, D. Divsalar, M.K. Simon, and P.J. McLane. *Introduction to Trellis-Coded Modulation with Applications.* Macmillan Pub Co, 1991.

[186] P. Elias. Coding for Two Noisy Channels. *Information Theory, The 3rd London Symposium*, pages 61–76, September 1955.

[187] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., 1991.

[188] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. Efficient Erasure Correcting Codes. *IEEE Transaction on Information Theory*, 47(2):569–584, Febbruary 2001.

[189] R.G. Gallager. *Low Density Parity Check Codes.* Monograph, M.I.T. Press, 1963.

[190] L. Rizzo. Effective Erasure Codes for Reliable Computer Communication Protocols. *ACM SIGCOMM Computer Communication Review*, 27(2):24–36, April 1997.

[191] D. Mackay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[192] M. Luby. LT Codes. *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS 02)*, pages 271–281, November 2002.

[193] A. Shokrollahi. Raptor Codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, June 2006.

[194] A. Varga. The OMNET++ Discrete Event Simulation System. *Proceedings of the European Simulation Multiconference (ESM 01)*, pages 319–324, June 2001.

[195] A. Varga and R. Hornig. An Overview of the OMNeT++ Simulation Environment. *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, March 2008.

[196] B. Zeigler. *Object-oriented Simulation with Hierarchical, Modular Models.* Academic Press, 1990.

[197] D. Dittrich. The "Tribe Flood Network" Distributed Denial of Service Attack Tool. *staff.washington.edu/dittrich/misc/tfn.analysis*, October 1999.

[198] C. Zou, W. Gong, and D. Towsley. Code Red Worm Propagation Modeling and Analysis. *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 138–147, November 2002.