A Survey of Fast-Recovery Mechanisms in Packet-Switched Networks

Marco Chiesa, Andrzej Kamisiński, Jacek Rak, *Senior Member, IEEE*, Gábor Rétvári, and Stefan Schmid

***Abstract*—In order to meet their stringent dependability requirements, most modern packet-switched communication networks support fast-recovery mechanisms in the data plane. While reactions to failures in the data plane can be significantly faster compared to control plane mechanisms, implementing fast recovery in the data plane is challenging, and has recently received much attention in the literature. This survey presents a systematic, tutorial-like overview of packet-based fast-recovery mechanisms in the data plane, focusing on *concepts* but structured around different networking technologies, from traditional link-layer and IP-based mechanisms, over BGP and MPLS to emerging software-defined networks and programmable data planes. We examine the evolution of fast-recovery standards and mechanisms over time, and identify and discuss the fundamental principles and algorithms underlying different mechanisms. We then present a taxonomy of the state of the art, summarize the main lessons learned, and propose a few concrete future directions.**

***Index Terms*—Fast reroute, network resilience, data plane.**

I. Introduction

PACKET-SWITCHED communication networks (datacenter networks, enterprise networks, the Internet, etc.) have become a critical backbone of our digital society. Today, many applications, e.g., related to health, business, science, or social networking, require always-on network connectivity and hence critically depend on the availability and performance of the communication infrastructure. Over the last years, several network issues were reported that led to major Internet outages in Asia [1], resulted in huge losses in revenues [2], affected thousands of airline passengers [3], or even disrupted the emergency network [4]. Many applications already suffer from small delays: A 2017 Akamai study shows that every 100 millisecond delay in website load time can lead to a significant drop in sales [5], and voice communication has a tolerable delay of less than 150 ms; for games it is often less than 80 ms [6].

In order to meet their stringent dependability requirements, communication networks need to be able to deal with *failures* which are becoming more likely with increasing network scale [7]. Today, link failures are by far the most frequent failures in a network [8], [9], and handling failures is a fundamental task of any routing scheme.

This paper focuses on packet-switched networks, where resilience to failures on the network layer can either be implemented in the so-called *control plane* or in thee so-called the *data plane*. In a nutshell, the control plane relies on global network information and is responsible for determining the paths along which packets are sent; the data plane is responsible for the logic of an individual switch, and in particular, the functions related to forwarding packets/frames from one interface to another, based on control plane logic. This separation of concerns enables a simple and fast data plane processing. Historically, resilience to network failures was implemented in the control plane: ensuring connectivity was considered the responsibility of the control plane while the data plane was responsible for forwarding packets at line-speed. Widely deployed routing schemes like OSPF [10] and IS-IS [11], hence include control plane mechanisms which leverage global message exchanges and computation to determine how to recover from link failures.

However, the slow reaction times of control plane mechanisms is becoming increasingly unacceptable [12]–[14]. Indeed, many studies have shown that control plane-based resilience can noticeably impact performance [7], also because of the high path re-computation time [15]. In basic solutions where the network can recover from failure only after the control plane has computed a new set of paths and installed the associated state in all routers, the disparity in timescales between packet forwarding in the data plane (which can be less than a microsecond) and control plane convergence (which can be as high as hundreds of milliseconds) can lead to long outages [16]. While recent centralized routing solutions based

Manuscript received June 28, 2020; revised December 12, 2020; accepted February 5, 2021. Date of publication March 11, 2021; date of current version May 21, 2021. This work was supported in part by COST Action CA15127 ("Resilient communication services protecting end-user applications from disaster-based failures" — RECODIS), supported by COST (European Cooperation in Science and Technology); *http://www.cost.eu*; in part by the Vienna Science and Technology Fund (WWTF) Project, Fast and Quantitative What-If Analysis for Dependable Communication Networks (WHATIF, 2020–2024) under Grant ICT19-045; in part by the KTH Digital Futures Center; and in part by Ericsson Research, Hungary. The work of Gábor Rétvári and Stefan Schmid was supported by the Austrian–Hungarian Joint Research Project FWF-30668/OTKA-135606. *(Corresponding author: Andrzej Kamisiński.)*

Marco Chiesa is with the Department of Computer Science, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden.

Andrzej Kamisiński is with the Department of Telecommunications, AGH University of Science and Technology, 30-059 Kraków, Poland (e-mail: andrzejk@agh.edu.pl).

Jacek Rak is with the Department of Computer Communications, Gdańsk University of Technology, 80-233 Gdańsk, Poland.

Gábor Rétvári is with the MTA–BME Information Systems Research Group and MTA–BME Momentum Network Softwarization Research Group, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, 1111 Budapest, Hungary.

Stefan Schmid is with the Faculty of Computer Science, University of Vienna, 1090 Vienna, Austria.

Digital Object Identifier 10.1109/COMST.2021.3063980

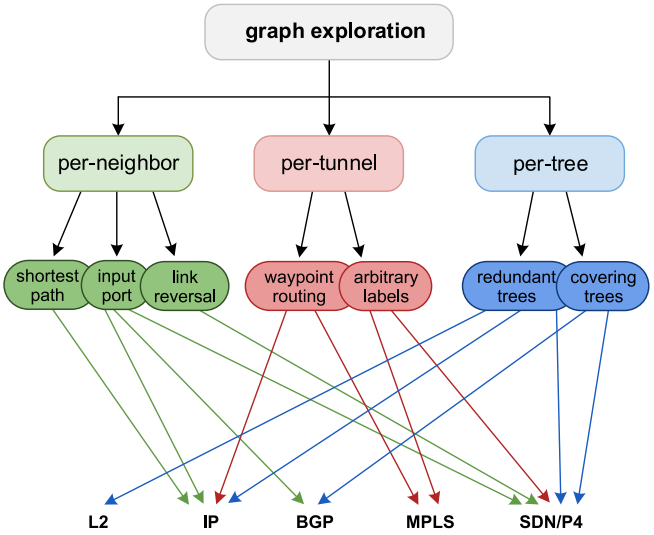

Fig. 1. A simplified classification of fast-recovery concepts and their mechanisms, as they appear on the different network layers.

on Software-Defined Networks (SDNs) [17]–[19], where all routing computation is performed by a controller which then pushes the results to the affected routers, are faster, there is still an inevitable delay of at least the round trip time between the routers and the controller.

Motivated by these performance issues, we currently witness a trend to reconsider the traditional separation of concerns between the control plane and the data plane of a network. In particular, given that the data plane typically operates at timescales several orders of magnitude shorter than the control plane [16], [20], moving the responsibility for connectivity to the data plane where failure recovery can in principle occur at the speed of packet forwarding is attractive.

Indeed, most modern networks support different kinds of *fast-reroute (FRR)* mechanisms which leverage *pre-computed* alternative paths at any node towards any destination. When a node locally detects a failed link or port, it can autonomously remove the corresponding entries from the forwarding table and continue using the remaining next hops for forwarding packets: a fast local reaction [21]. In FRR, the control plane is hence just responsible for *pre-computing* the failover paths; when a failure occurs, the data plane utilizes this additional state to forward packets. For example, many data centers use Equal Cost MultiPath (ECMP) [22] (a data plane algorithm that provides automatic failover to another shortest path), WAN networks leverage *IP Fast Reroute* [23]–[25] or *MPLS Fast Reroute* [26] to deal with failures on the data plane, SDNs provide FRR functionality in terms of *OpenFlow fast-failover groups* [27], and BGP relies on BGP-PIC [28] for quickly rerouting flows, to just name a few.

Implementing FRR mechanisms, however, is challenging and requires careful configuration, as the failover behavior needs to be *pre-defined*, before the actual failures are known. Additional challenges are introduced due to the limited functionality in the data plane as well as the stringent latency requirements which do not allow for sophisticated reactions. Configuring FRR is particularly tricky under multiple and correlated failures [29]–[33]. FRR mechanisms in the data plane are hence often seen as a "first line of defense" and lack guarantees: they support a fast but perhaps suboptimal reaction (due to the limited information about the actual failure scenario). In a second phase, the control plane may re-establish the routes more effectively, meeting advanced traffic engineering criteria.

A. Our Contributions

Motivated by trend of moving resilience mechanisms in packet-switched networks to the data plane to speed up reaction times, this paper provides a structured survey of fast-recovery mechanisms. Our primary goal is to familiarize the reader with selected concepts, in a tutorial-like manner, which together form a representative set of fast-recovery methods in the data plane. Indeed, as we will see, different approaches have different advantages and disadvantages, and a conceptual understanding is hence useful when reasoning about which technology to deploy. The topic is timely, not only because of the increasing dependability requirements on communication networks and the resulting need to move the responsibility for connectivity to data plane mechanisms, but also due to the emergence of programmable data planes which introduce new functionality in the data plane, allowing to implement different approaches. We believe that this provides an opportunity for a structured survey.

To provide a systematic understanding, we structure our discussion around the underlying technologies (e.g., Ethernet, MPLS, IP, SDN) as well as the related use cases (e.g., failure scenarios, inter-domain routing, intra-domain routing, data centers) and technological constraints. We then highlight algorithmic aspects (e.g., complexity) and performance issues (e.g., the total time needed to complete the recovery process). Throughout the paper, we provide clear explanations of the related technical terms and operation principles and use illustrations based on simple examples.

Specifically, we show that, in some dimensions, existing approaches can differ significantly. For example, timing requirements on recovery on Layer-2 (tens of microseconds) are orders-of-magnitude different than on the IP layer (tens of milliseconds), which are in turn orders-of-magnitude different than on BGP (seconds). But also technological constraints can differ significantly, and what works in SDN may not work in IP or BGP; further constraints arise in terms of hardware support. Nevertheless this survey will also highlight the numerous connections that exist between the different approaches and layers, and how concepts influenced each other.

Fig. 1 shows an example of some major concepts used for fast recovery on the different network layers. On a high level, ensuring that a destination can be reached even under multiple failures can be seen as some kind of graph exploration problem. We observe that existing solutions in the literature can roughly be classified into three categories, depending on whether the local exploration strategy is defined per-neighbor, per-tunnel or per-tree. For each of these categories, different

algorithmic concepts are used, which re-appear on the different networking layers, sometimes in different flavors.

In summary, this paper is the first to overview fast-recovery mechanisms in the data plane from the perspective of a comprehensive set of available technologies mentioned above as well as a broad range of characteristics, including:

- the underlying general concepts (e.g., loop-free alternate next-hop, input interface-aware routing, additional/extended FIBs and other data structures, encapsulation/tunneling, redundant spanning trees, failure coverage analysis and improvement);
- improvements to other solutions, including the motivation for the selected changes and the illustration of the evolution of ideas;
- the selected algorithmic aspects, performance;
- the maximum number of simultaneous failures handled by the scheme;
- mode of operation (i.e., distributed, centralized);
- signaling aspects (whether the use of packet header/dedicated control messages for signaling is needed, or no signaling is required);
- reliance on existing routing protocols;
- technological constraints;
- deployment considerations including the required modifications in the data/control plane, gradual deployment, possible difficulties;
- the use cases presented by the proposers of schemes and the related assumptions/requirements (time scale, convergence, or investigated failure models).

B. Scope, Novelty, and Target Audience

This paper is motivated by the rising importance and popularity of fast-recovery mechanisms, as well as the emergence of software-defined and programmable networks, which enable innovative FRR approaches. Our focus is hence on data plane mechanisms, and in particular, packet-based fast-recovery mechanisms, and we do not discuss control plane mechanisms nor the orthogonal issue of how to *detect* failures in the first place. We concentrate on the most common case, the case of simple best-effort unicast packet communication, and we barely touch on issues related to multicast, QoS, and traffic engineering. Our focus is on wired networks and, due to the special requirements, we leave the discussion of fast recovery in wireless networks for a future survey. Furthermore, given the focus on concepts and the tutorial-style nature of this article, we do not aim to provide a complete survey of the research literature, which is beyond the scope of this paper.

With our focus on packet-switched networks we hence explicitly do not consider recovery in, e.g., optical networks for which there exists much equipment for performing protection. Furthermore, there already are many good surveys on reliable routing in specific communication technologies, such as Ethernet [34], IP [23], [25], [35], [36] (and more recently [23], [25]), MPLS [37], [38], BGP [39], or SDN [40]. However, to the best of our knowledge, this is the first complete and up-to-date survey on the timely topic of fast-recovery mechanisms in packet-switched networks. We believe that only such an overarching approach can highlight the conceptual similarities and differences, and hence help choose the best approach for a specific context, which is particularly relevant for emerging networks. Indeed, our goal is *not* to provide a comprehensive survey of the literature. Rather, in order to provide an understanding of the underlying key concepts, our paper focuses on the fundamental mechanisms across a spectrum of different technologies and layers.

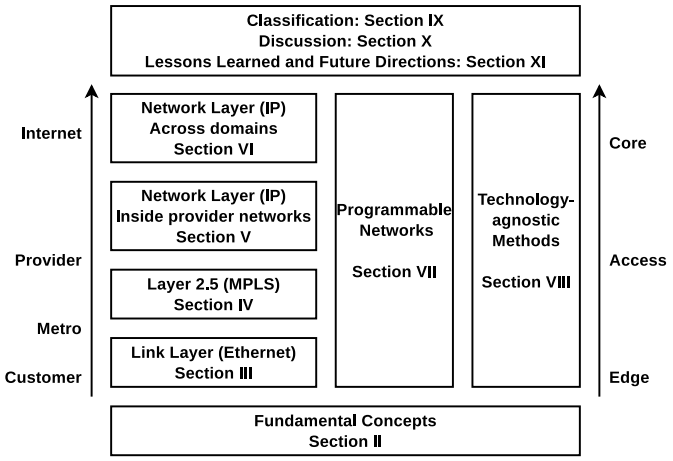

Fig. 2. Organization of the paper.

Our paper hence targets students, researchers, experts, and decision-makers in the networking industry as well as interested laymen.

C. Organization

The organization of this paper adopts the traditional protocol-layer-based view of packet-switched communication networks. As mentioned above, the perspective in this paper is motivated by emerging technologies such as SDNs and programmable data planes which bring together and unify specific layers and technologies. For this, however, we first need an understanding of the technologies on the individual layers. Therefore, after Section II, which provides a through overview of the fundamental concepts related to network resilience, in each section we review the most important ideas, methods, and standards related to the different layers in the Internet protocol stack in a bottom-up order (see Fig. 2). Whenever a protocol layer spans multiple operational domains, the organization follows the traditional edge/access/core separation principle. Inside each section then, the review takes a strict chronological order, encompassing more than 25 years of developments in data-plane fast-recovery methods.

We start with the link-layer (in Section III) which is an intrinsic and crucial part of the packet-switched protocol stack and our first line of defense when it comes to failures, and mostly used in local and metro area networks. Interestingly, to the best of our knowledge, despite the topic's importance, the link-layer FRR has not been surveyed yet. Then, Section IV reviews fast recovery for MPLS, operating as the "Layer 2.5" in the Internet protocol stack, deployed in access networks and, increasingly, in provider core networks. Section V presents IP Fast ReRoute, the umbrella network-layer fast-recovery framework designed for a single operator domain (or Autonomous

TABLE I
CLASSES OF SERVICE DEFINED BY ITU-T

Class of Service	Description of applications	IPTD	IPDV	IPLR	IPER
Class 0	Real-time, jitter–sensitive, highly interactive (e.g., VoIP, video teleconference)	100 ms	50 ms	1×10^{-3}	1×10^{-4}
Class 1	Real-time, jitter–sensitive, interactive (e.g., VoIP, video teleconference)	400 ms	50 ms	1×10^{-3}	1×10^{-4}
Class 2	Transaction data, highly interactive (e.g., signaling)	100 ms	undefined	1×10^{-3}	1×10^{-4}
Class 3	Transaction data, interactive	400 ms	undefined	1×10^{-3}	1×10^{-4}
Class 4	Tolerating low loss (e.g., short transactions, bulk data, video streaming)	1 s	undefined	1×10^{-3}	1×10^{-4}
Class 5	Typical applications of IP networks	undefined	undefined	undefined	undefined

System), and then Section VI discusses the selected fast-recovery solutions designed for wide-area networks and, in particular, the Internet core. The separation emphasizes the fundamental differences between the cases when we are in full control of the network (the intra-domain case, Section V) and when we are not (Section VI). Next, we review the methods that do not fit into the traditional layer-based view: Section VII discusses the fast-recovery mechanisms proposed for the emerging Software-Defined Networking paradigm, while Section VIII summarizes generic, technology-agnostic solutions.

Finally, we cast the existing fast-recovery proposals in a common framework and discuss some crucial related concepts. In particular, in Section IX we give a comprehensive family of taxonomies to classify fast-recovery schemes along several dimensions, including data-plane requirements, operation mode, and resiliency guarantees. In Section X we briefly discuss some critical issues regarding the interaction of the control-plane and the data-plane schemes during and after a recovery. Last, in Section XI we conclude the paper and identify the most compelling open issues for future research.

II. FUNDAMENTAL CONCEPTS

The undoubtedly large amount of data transmitted by communication networks makes the need to assure their fault-tolerance one of the fundamental design issues. Among a variety of reasons for single and multiple outage scenarios of network nodes or links, failures of single physical links are dominant events in wide-area networks [41]. They are mostly a result of random activities such as a cable cut by a digger. According to [42], [43], there is one failure every 12 months related to each 450 km of links, while the average repair time of a failed element is 12 hours. According to recent statistics, the frequency of multiple failures (i.e., simultaneous failures of multiple nods and links) — especially those occurring in a correlated manner, e.g., as a result of malicious human activities, or forces of Nature (tornadoes, hurricanes, earthquakes, etc.) — is raising [44], [45]. This in turn makes it challenging to assure high availability of network services, since routing protocols typically used in IP networks (such as Border Gateway Protocol — BGP [46] or Open Shortest Path First – OSPF [10]) are characterized by a slow process of a post-failure routing convergence which can even take tens of seconds [42], [47].

The behavior of conventional routing protocols can thus be acceptable only for delay-tolerant applications, provided that the failures are indeed not frequent. However, for a wide range of applications with stringent QoS requirements highlighted in [48], [49] and summarized in Table I, such a slow convergence is unacceptable. In particular, concerning the set of four intrinsic parameters (i.e., related to network performance) of Quality of Service — QoS [48], [50], [51] defined for IP networks as: the maximum IP Packet Transfer Delay — IPTD, IP Delay Variation — IPDV, IP packet Loss Ratio — IPLR and IP packet Error Ratio — IPER (see ITU-T recommendations Y.1540 and Y.1541 in [52], [53]). As shown in Table I, for applications belonging to the first four Classes of Service — CoS, (i.e., classes 0–3), the considered the values of transmission delay undoubtedly impacted after a failure should not be higher than 100–400 ms. Similarly, the values of delay variation for classes 0–1 should be at most 50 ms, which during the network recovery phase is even around two-three orders of magnitude less than what the conventional IP routing protocols can offer.

It is important to note that the majority of failure events in IP networks is transient [47], [54], [55] and, therefore, may lead to a lack of routing convergence during a significant time. Mechanisms of fast recovery of flows are thus crucial to assure ***network resilience***, i.e., the ability to provide and maintain an acceptable level of service in the presence of various faults and challenges to normal operation [56]–[60].

To explain the fundamental concepts of network resilience to assure fast recovery of the affected flows, in Section II-A, we first present the set of relevant disciplines of network resilience followed by definitions of resilience measures. In particular, the group of characteristics analyzed in Section II-A is essential for evaluation of fast recovery which we define in this paper as the ability of a network to recover from failures according to the time-constrained recovery plan to meet the QoS-related requirements of applications. Next, Section II-B provides a detailed taxonomy of network resilience mechanisms based on the utilization of the alternate paths with a particular focus on service recovery time. Finally, in Section II-C, information on the steps of the network recovery procedure is given.

A. Disciplines and Measures

As discussed in [56], resilience disciplines can be broadly divided into two categories: *challenge tolerance* comprising the design approaches to assure the continuity of service and *trustworthiness* focusing on the measurable characteristics of network resilience shown in Fig. 3.

Following [56], challenge tolerance includes disciplines addressing the network design issues to provide service

TABLE II
SELECTED ITU-T RESILIENCE METRICS

ID	Area	Metric
E.800	General (e.g., Internet access)	**Instantneous (un)availability –** probability for a network element of being in an `up` (`down`) state at a given time
E.802		**Mean time between failures (MTBF) –** mean time between two consecutive failures of a repaired element
E.820		**Mean time between interruptions (MTBI) –** mean time between the end of one interruption and start of the next one
E.850		**Mean time to failure (MTTF) –** mean value of time since the last change of state from `down` to `up` until the next failure
E.855		**Mean time to recovery (MTTR) –** mean value of time when a network element is in `down` state due to a failure
E.860		**Mean up time / mean down time (MUT/MDT) –** mean value of time when a network element is in `up`/`down` state
E.862		**Reliability function *R*(*t*) –** probability for a network element of bein in `up` state in (0, *t*) interval
E.880		**Retainability –** probability that a service will continue to be provided
Y.1540	IP	**IP packet loss ratio (IPLR) –** the total number of lost packets divided by the total number of transmitted packets
Y.1541		**Service availabiliy –** a share of the total service time classified as available using the threshold on IPLR
Y.1542		**IP service (un)availability (PIU/PIA) –** part of time of (un)available IP service based on IP service (un)availability function
Y.1561	MPLS	**Packet loss ratio (PLR) –** similar to IPLR
		Severe loss block (SLB) – an event at the ingress node for a block of packets with packet loss ratio above the upper bound
		Recovery time – time for recovery operations based on the number of successive time intervals of SLB outcomes
		Service availability, PIU, PIA – defined in a similar way as in Y.1540 but in the context of SLBs
Y.1562	Higher layer protocols	**Service availability –** similar to Y.1540 but related to the transfer delay and service success ratio

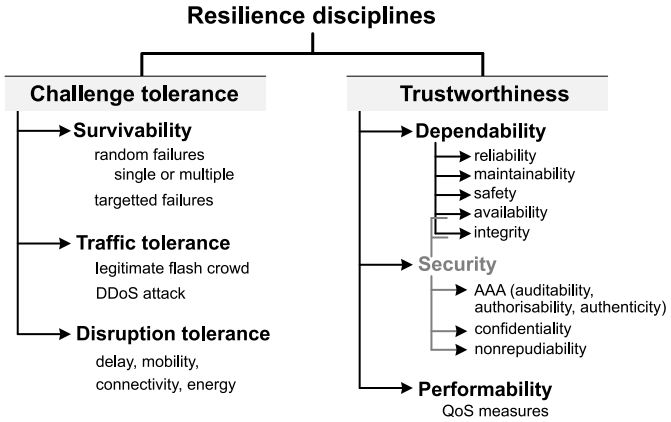

Fig. 3. Classification of network resilience disciplines based on [56].

continuity in the presence of challenges. Among them, *survivability* denotes the capability of a system to fulfil its mission in the presence of threats, including natural disasters and attacks. *Traffic tolerance* refers to the ability to tolerate the unpredictable traffic. It may be a remarkable challenge in a multiple-failure scenario implied, e.g., by a disaster (such as a tsunami after an earthquake [61], [62]), or in other situations including, e.g., DoS attacks where the traffic volume is raised unexpectedly far over the expected peak value for the normal operational state [45]. *Disruption tolerance* focuses on the aspects of connectivity among network components primarily in the context of nodes mobility, energy/power challenges, or weak/episodic channel connectivity [63].

Trustworthiness, in turn, comprises the measurable characteristics of the resilience of communication systems to evaluate the assurance that the system will perform as expected [64]. It includes three disciplines, namely: dependability, security and performability. *Dependability* is meant to quantify the level of service reliance. It includes:

- *reliability* $R(t)$ being a measure of service continuity (i.e., probablity that a system remains operable in a given $(0, t)$ time period,
- *availability* $A(t)$ defined as the probability that a system is operable at time t.
  Its particular version is the *steady-state availability* defined as a fraction of a system lifetime during which the system is accessible, as given in Eq. (1).

$$A = \frac{MTTF}{MTTF + MTTR} \tag{1}$$

  where:
  MTTF denotes the mean time to failure,
  MTTR is the mean time to repair.
- *maintainability* being the predisposition of a system to updates/evolution,
- *safety* being a measure of system dependability under failures,
- *integrity* denoting protection against improper alterations of a system.

Security denotes the ability of a system to protect itself from unauthorized activities. It is characterized by both joint properties with dependability (i.e., by availability, and integrity) as well as individual features including authorisability, auditability, confidentiality, and nonrepudiability [65]. *Performability* provides measures of system performance concerning the Quality of Service requirements defined in terms of transmission delay, delay variation (jitter), throughput/goodput, and packet delivery ratio [56].

As the impact of resilience on the Quality of Service is evident, a concept of *Quality of Resilience* (*QoR*) has been introduced in [66] to refer to service resilience characteristics. In contrast to QoS characteristics being relatively short-term, the majority of attributes of resilience relating to service continuity, downtime, or availability are long-term by nature [66]. Indeed, most of resilience metrics proposed by International Telecommunication Union–Telecommunication Standardization Sector (ITU-T) summarized in [66] and shown

in Table II can only be measured in the long term based on end-to-end evaluations. It is also worth noting that, unlike many QoS measures, QoR characteristics cannot be perceived by users in a direct way, for whom it is not possible to distinguish between a network element failure and network congestion when noticing the increased transmission delay/packet losses.

Despite the existence of a number of resilience measures, in practice only two of them are widely used, i.e., the mean time to recovery (MTTR) and the steady-state availability defined by Eq. (1), which is also impacted by the recovery time (MTTR factor) [66]. Therefore, in the following part of this section presenting the taxonomy of recovery schemes and recovery procedure, a particular focus in these parts is on the recovery time issues.

B. Taxonomy of Recovery Methods

The need for fast rerouting has its roots in an undoubtedly slow process of post-failure routing convergence of contemporary schemes such as BGP or OSPF which can even take tens of seconds. The key objective is to reduce the convergence time to the level of less than several tens of milliseconds [35]. In this context, IP network resilience is often associated with the path-oriented Multiprotocol Label Switching (MPLS)-based recovery schemes. IP-MPLS Recovery mechanisms aim to redirect the traffic served by the affected working paths onto the respective alternate (backup) routes [66].

A common observation is that differentiated resilience requirements characterize different traffic flows. Therefore, to prevent the excessive use of network resources, it is reasonable to decide on the application of specific recovery schemes on a per-flow basis [67]. In particular, this would mean that only those flows requiring a certain level of service availability in a post-failure period need to be provided with a recovery mechanism.

Following [58], [68], recovery methods can be classified based on several criteria, the most important ones including the backup path setup method, the scope of the recovery procedure, the usage of recovery resources, the domain of recovery operations, or the layer of recovery operations shown in Fig. 4.

Concerning the backup path set up method, the alternate paths can be either configured in advance (pre-computed) at the time of setting up the primary path (known as *preplanned* (*protection*) switching scheme, or established dynamically after detection of a failure (referred to as the *restoration/rerouting* concept) [67], [69]. Preplanned protection provides faster recovery of the affected flows as alternate paths are pre-established before the failure. However, a disadvantage is its high cost due to the increased level of resource consumption for backup paths set up well before the failure and often used only for a relatively short post-failure period. Restoration methods in practice are considered as a default solution for current IP networks [66]. They are remarkably more capacity-efficient but do not provide 100% of restorability, as the appropriate network resources are not committed before a failure and may not be available for the alternate paths after a failure. Another disadvantage of the restoration schemes

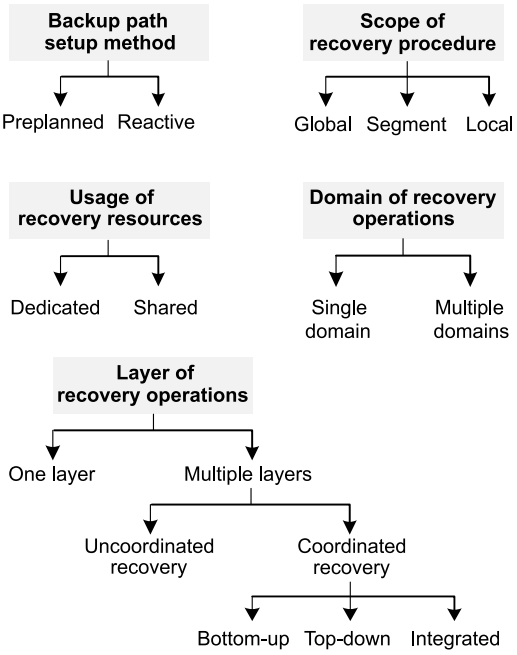

Fig. 4. Classification of recovery methods.

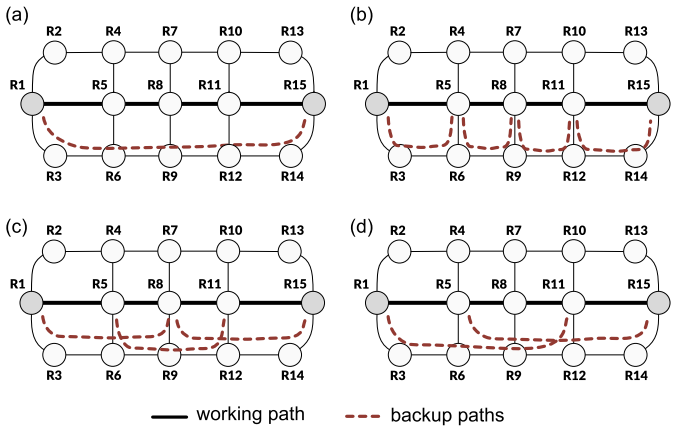

Fig. 5. Scope of the backup path: (a) path protection, (b) local protection against a link failure, (c) local protection against a node failure, (d) segment protection.

is their lower time-efficiency, as recovery procedure in their case also involves the phase of a backup path calculation.

Protection switching approaches based on preplanned protection are reasonable for IP flows requiring service recovery time below 100 ms [67]. For flows with restoration time between 100 ms and 1 s, it is appropriate to apply a restoration scheme such as MPLS restoration [70]. In the context of the other flows able to tolerate the recovery switching time over 1 s, a conventional Layer 3 rerouting is commonly used. Flows with no resilience requirements are usually not recovered, and their resources are freed (i.e., preempted) to enable recovery of other flows with resilience requirements [67].

Concerning the scope of a backup path [69], [71], recovery schemes can be divided into:

– *global* schemes where a single backup path protects the entire working path (Fig. 5a). Global schemes are commonly associated with reservation of resources before

a failure (i.e., resources pre-reserved and alternate paths pre-established). Such backup paths can be used either after failure only (the so-called 1:1 path protection) or in parallel with the working path to carry the traffic in a normal state (1+1 protection model). Under path protection, switching the traffic onto a backup path is done at the ingress-egress pair of nodes (i.e., the end nodes of a path) as shown in Fig. 5a,

– *local* schemes (with backup paths being either set up in advance or dynamically after a failure) enabling short detours over the failed link of the affected path (Fig. 5b) or over two consecutive links in the case of a failure of a node (Fig. 5c),
– *segment* schemes with backup paths protecting certain segments of backup paths (Fig. 5d).

A general observation is that with a decrease of the scope of backup path protection (i.e., from global toward local protection schemes), the time of service restoration decreases. It can be explained by the fact that for local protection schemes the redirection of the affected flows onto the backup paths is done closer to the failure as well as because backup paths under local protection are remarkably shorter than the respective ones for global protection. Therefore, recovery schemes based on local detours (especially local protection methods involving pre-reservation of backup path resources) are often referred to in the literature as *Fast-Reroute* concepts [71].

However, fast switching of flows onto backup paths for local protection schemes is achieved at the increased ratio of network redundancy (denoting the capacity needed to establish backup paths) when compared to global protection schemes. It can be justified by the fact that the total length of backup paths protecting a given primary path for local protection is greater than the length of a backup path assuring the end-to-end protection in the global protection scheme. Local recovery schemes are therefore seen as fast but often more expensive (in the case of local protection) than the respective global schemes, which are, in turn, more capacity-efficient, easier to optimize, but slower concerning recovery time [66].

Network resources (capacity of links) reserved for backup paths can be either dedicated (i.e., reserved for those backup paths exclusively), or *shared* among a set of backup paths. To provide the network resources for backup paths after a failure, sharing the link capacity by a set of backup paths is possible if these backup paths protect mutually disjoint parts of working paths (i.e., to guarantee that after a failure, there would be a need to activate at most one of these backup paths) [58]. The use of dedicated backup paths may result in faster recovery, as dedicated backup paths can be fully pre-configured before a failure. However, it is undoubtedly expensive, as backup paths often require even more network resources than the corresponding working paths (by default, they are longer than the parts of working paths they protect). Another observation is that a classification of backup resource reservation schemes into dedicated and shared is characteristic strictly to protection methods [42]. Major schemes of reactive recovery, in turn, involve reservation of just dedicated resources for the alternate paths (as these paths are merely the only ones operating after a failure).

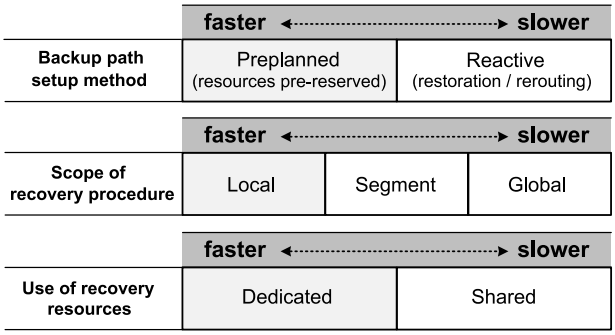

Fig. 6. Relations among the recovery methods for IP networks concerning the recovery time.

Fig. 6 presents relations among the recovery methods for IP networks discussed in this section concerning the time of recovery.

For networks divided into multiple domains (each domain often managed by a different owner), recovery actions are performed in these domains separately. In single-domain networks, one recovery scheme can be, in turn, applied in an end-to-end manner for all flows.

Architectures of communication networks are inherently multilayer meaning that one transmission technology such as Optical Transport Network (OTN) using optical (Wavelength Division Multiplexing – WDM) links serves as a carrier for another transfer architecture such as, e.g., IP network [42], [72]. Failures seen by the upper layer can thus happen originally at the lower layer(s) making network recovery a challenging issue [47]. It is also worth noting that any failure in the lower layer, if not handled on time by the lower layer, may manifest itself as a simultaneous failure of a number of elements in the upper layer [67]. Available techniques of multi-layer recovery are classified based on the sequence of recovery operations as either bottom-up, top-down, or integrated. As a detailed analysis of multi-layer recovery schemes is outside the scope of this paper, the reader is referred to [73] for more details.

C. Recovery Procedure

As explained in [66], [68], [70], [71] there are seven consecutive phases of recovery of the affected IP flows in a period between the occurrence of a failure and the physical repair of a failed network element shown in Fig. 7. They include fault detection, fault localization, hold-off period, fault notification, recovery switching, restoration completion, and normalization.

The objective of the *fault detection* phase is to notice the failure in the network (time T_1 in Fig. 7). A failure can be detected either by a management or a transport (data) plane [42], [74], e.g., at the level of optical paths forming the IP virtual links or in the IP network.

Concerning the management plane, failures can be identified by network elements close to the faulty item using the Loss of Clock, Loss of Modulation, Loss of Signal, or degradation of signal quality (e.g., increased signal-to-noise ratio – SNR) [68]. For instance, failure detection in optical networks makes use of information on the optical power or the temperature at the transmitter, or the input optical power at the

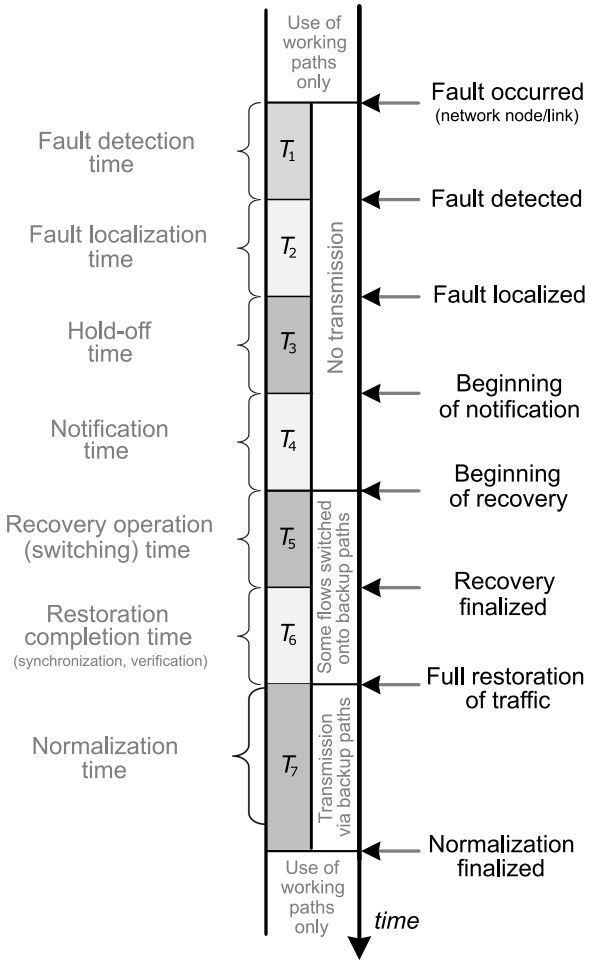

Fig. 7. Restoration time components based on [66], [68], [70], [71].

receiver, power distribution of carriers over the full bandwidth/channel wavelength, or crosstalks [75]. Other hardware components which can generate alarms in optical networks include optical regenerators/reshapers/retimers – 3Rs (e.g., when it is not possible for them to lock to the incoming signal) or switches when they cannot establish a new connection [76].

In the data plane, a fault can be detected by observing a degraded quality in the context of an increased Bit Error Ratio – BER, e.g., by CRC computation (Ethernet), TCP/IP checksum verification [75], and by noticing the increased values of the end-to-end quality parameters such as lower throughput or increased data transmission delay [42]. In multilayer networks, the upper-layer algorithms referring to failure detection in IP and overlay networks can be broadly classified into active and passive schemes [77]. In the active schemes, periodic keep-alive messages are exchanged between neighbouring nodes. In this case, fast detection of failures comes at the price of an increased amount of control overhead. One of the related examples is Bidirectional Forwarding Detection (BFD) mechanism [78]. On the other hand, passive schemes only make use of data packet delivery to confirm correct operation of nodes and links (however, if data packets are not sent frequently enough, passive methods are hardly useful). The status of a given node can then be determined by other nodes either independently, or collaboratively [79].

It is worth noting that, failure detection in multilayer networks is often one of the most redundant tasks, as it is commonly performed at multiple layers [76].

Fault localization (represented by time T_2 in Fig. 7) means identification of the faulty element (point of failure) necessary to determine the network element at which the transmission should be suspended [42], [68]. Precise localization of a faulty element as well as identification of the element type (node/link) is crucial especially in the context of local repair methods, where redirection of the affected flows is performed just around the failed node/link.

In general, fault localization is considered as one of the most difficult tasks. It is commonly based on the observations of symptoms which can help to infer a precise location of a failure in the network. As presented in [74], fault localization techniques can be broadly classified into passive and active methods. Techniques of passive monitoring wait passively until the respective alarms are received from the monitoring agents installed on the network elements to report the failures. Passive schemes can be further classified into AI-based (using the artificial intelligence expert systems and a knowledge database), model-based (describing the system behaviour as a mathematical model), and graph-theoretic schemes (using graphical models of fault propagation). Techniques of active monitoring, in turn, utilize probing stations to generate special packets called probes (using, for instance, ping or traceroute) to monitor the network state actively [74].

In any layered structure, where the IP layer is typically considered as the uppermost one (as in the IP-over-WDM architecture [58], [73]), there is a need to decide on the sequence of layers to perform the recovery operations. In general, recovery operations at the lower (e.g., WDM) layer are executed at coarser granularity (due to the aggregation of IP flows onto optical lightpaths), which reduces the number of recovery operations. Also, if WDM recovery operations are performed fast enough, they can be entirely transparent to the IP layer. Only those failures that cannot be recovered in the WDM layer (e.g., failures of IP routers) need to be handled in the IP layer. Therefore, in the context of the bottom-up sequence of recovery actions, the *hold-off* period (time T_3 in Fig. 7) is used to postpone the recovery operations in the IP layer (e.g., related to failures of the IP layer nodes) until the respective recovery operations are performed first by the lower (WDM) layer [73].

The objective of *fault notification* initiated by a node neighbouring to the faulty element is to inform the intermediate nodes along the primary path about a failure and the end nodes of the protected segment of the working path (referred to as ingress and egress nodes in the case of global/segment schemes) to trigger the activation of the backup path. It is essential to notice that fault notification time (T_4 in Fig. 7) can be neglected for local repair schemes such as link protection. For example, as illustrated in Fig. 5c), the node detecting the failure (i.e., the ingress node of the affected part of a working path) is the one to redirect the affected flow.

During *recovery operation (switching)* interval (time T_5 in Fig. 7), reconfiguration of switching at network nodes takes place to redirect the affected flows onto the respective

alternate paths. This stage is more time-consuming for restoration schemes than for protection strategies, as it also includes calculation and installation of the alternate path [71].

The recovery procedure is completed after the verification and synchronization phase (given by time T_6 in Fig. 7) when the alternate path is verified and synchronized, as well as after the traffic is next propagated via the alternate route and reaches the end node of the backup path [42], [71].

After the traffic is switched onto backup paths, the process of manual repair of failed network nodes/links by the personnel is initiated. In practice may take at least several hours [42]. The objective of this *normalization* phase (T_7 in Fig. 7) is to restore the original network characteristics from the period before the failure. In particular, normalization also includes the need to update the communication paths, as the recovery paths used after a failure are non-optimal in the physically repaired network (these paths are commonly longer and use more network resources than the respective working paths do before a failure). Therefore, the normalization phase is also to revert the traffic onto the original set of communication paths utilized before the failure.

An important conclusion following from this subsection and the previous one is that fast recovery of the affected flows (especially essential for real-time services with stringent requirements on availability and reliability such as Voice-over-IP [54], [80]) is possible by the application of local protection schemes. These methods involve short backup paths installed before the failure and eliminate the need to send the end-to-end notifications along the working path [71]. Therefore, proactive IP fast-reroute schemes investigated in the remaining part of this paper are based on the local protection concept.

III. Link-Layer Fast Recovery

We start our protocol-layer-based considerations with the link layer, which plays an important role for the recovery of packet-switched networks and typically serves as a first line of defense when failures occur in local and metro area networks. In particular, in this section we analyze the characteristics of fast recovery mechanisms proposed for Ethernet – the major Layer-2 technology.

Resilient routing in Ethernet networks is challenging. On the one hand, fast-recovery mechanisms are necessary to minimize message losses and increased transmission delay due to one or more failures. However, this is challenging in Ethernet, as frames do not include a Time-to-Live (TTL)-like field known from the network-layer IPv4 protocol, and even transient forwarding loops can cause significant problems. The first major solution designed to avoid forwarding loops while providing basic restoration capabilities in Ethernet networks was the **IEEE 802.1D Spanning Tree Protocol (STP)** introduced in [81]. The main idea behind STP is to establish a single spanning tree across the network to assure that there is a unique path between any two nodes. However, despite being simple and easy to scale, it has not been designed to provide fast failure recovery. As mentioned in [82], its disadvantage is in a remarkably slow convergence time of even up to 50 s [83], which is not acceptable for many applications and gets magnified in networks consisting of hundreds of switches.

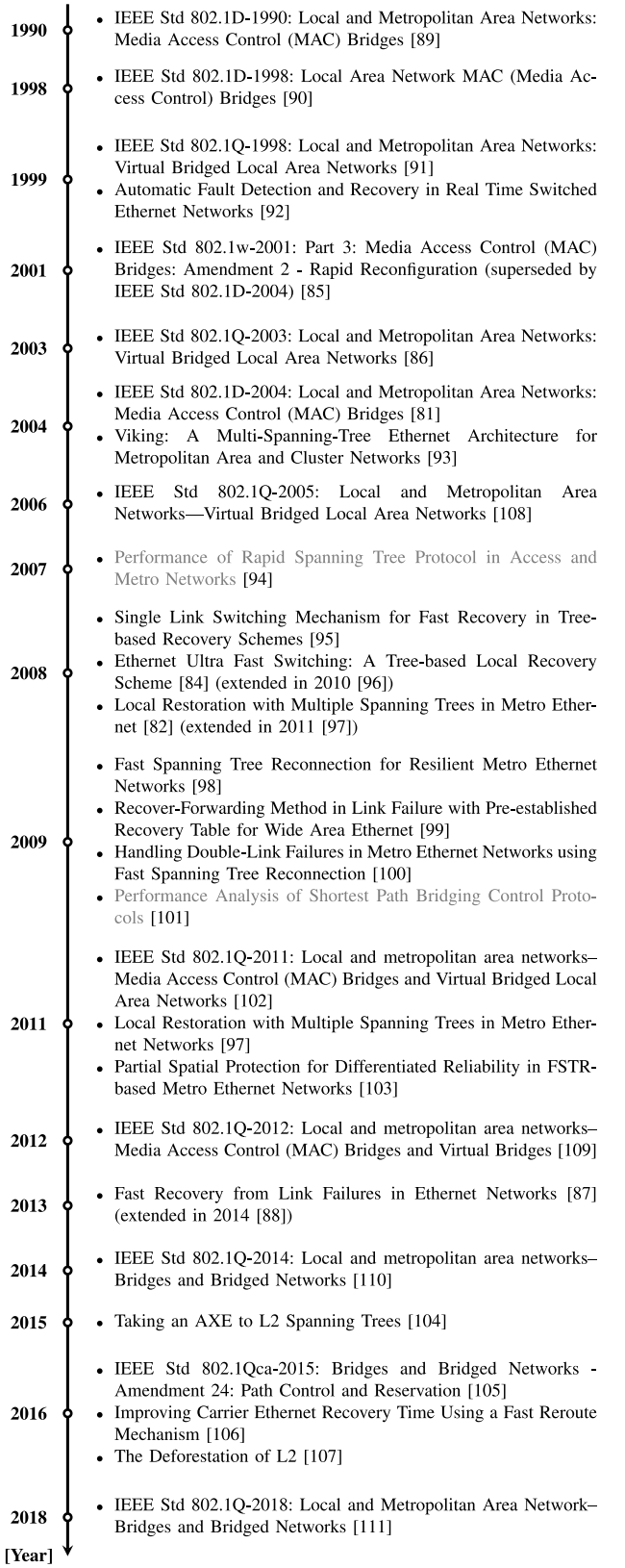

Fig. 8. Timeline of the selected documents and fast-recovery solutions operating at the link layer (entries marked in gray provide the general context).

Based on the scope of recovery, we can distinguish between *global recovery* schemes to protect any node/link except for the source/destination node of a demand, and *local recovery*

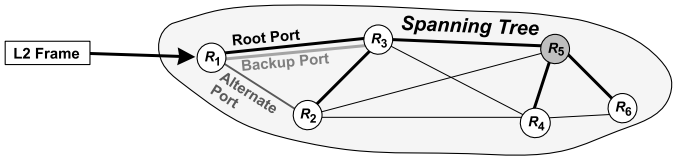

Fig. 9. Illustration of the example configuration of a spanning tree for RSTP rooted at node R_5: An incoming frame is forwarded by R_1 by default in the direction of the root node (R_5 here) via the respective *Root Port* outgoing from R_1 towards R_3. In the case of a failure of the primary link R_1–R_3, the frame can be forwarded along a duplicate link via the *Backup Port*. However if both direct links between R_1 and R_3 are not available (e.g., due to their failure), the frame can be forwarded via the *Alternate Port* towards node R_2.

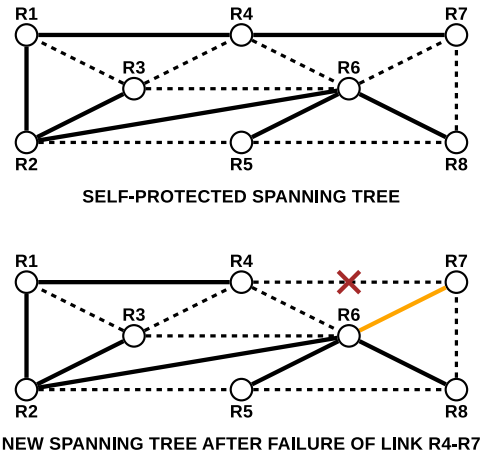

Fig. 10. Illustration of a self-protected spanning tree (thick black lines) and the new tree returned by the single link switching mechanism after a single link failure (the newly attached link is marked in orange).

approaches [84] to protect against a failure of the incident node/link, minimizing the time necessary for recovery. The global recovery is initiated by the source/destination node, while local recovery is triggered by the immediate upstream node of the failed network element.

Three major IEEE Ethernet spanning tree protocols are the Spanning Tree Protocol (STP), Rapid Spanning Tree Protocol (RSTP) [81], [85], and Multiple Spanning Trees Protocol (MSTP) [84], [86]. Among them, STP is considered to be the first spanning tree protocol for Ethernet with resilience functionality, which, upon a failure, triggers the spanning-tree reconstruction procedure. However, when using STP, the problem is that links which do not belong to the spanning tree cannot be used to forward traffic, which may lead to increased resource utilization and local link congestions in other areas of the network. Since the introduction of STP, several mechanisms have been proposed to solve this issue (see the related evolution timeline shown in Fig. 8). We discuss the selected representative examples in the following sections. For a discussion of different solutions related to optical networks, the reader is referred to [58], [68], [73].

A. Solutions Based on a Single Spanning Tree

The motivation for the introduction of the **Rapid Spanning Tree Protocol** (RSTP, IEEE 802.1D [81], [85]) was a reduction of a negative impact of a long convergence time of a single spanning tree on Ethernet network performance. It operates in a distributed fashion and relies on the proposal-agreement handshaking algorithm to provide synchronization of the state by switches. RSTP also introduces the concept of specific Port Roles related to recovery processes. In particular, as shown in Fig. 9, the *Alternate Port* and *Backup Port* roles are assigned to such ports of a bridge which can be used to provide connectivity in the event of failure of other network components [85].

Concerning the advantages of RSTP, it not only prevents forwarding loops but also enables transmission on the redundant links in the physical network topology. Due to the popularity of this protocol, it was evaluated by many research groups also in real test networks. For instance, in [94] it was evaluated in the context of access and metro Ethernet networks by investigating the failure detection time and additional delays introduced by hardware. The results reported in [94] confirm that RSTP can converge within milliseconds.

Another mechanism aimed at a reduction of the reconfiguration time of STP is proposed in [92]. The main idea behind this scheme is to avoid using conventional timeouts to inform about the local completion of a tree formation phase. Instead, the approach uses the additional explicit termination messages which are sent backwards from nodes to their parent nodes in the newly formed tree. Therefore, the scheme is able to converge in a shorter time (the evaluation results presented in [92] show that the tree recovery time can be remarkably reduced even to less than 50 ms for a moderate-size network).

Another way to provide fast recovery of the affected traffic is to reuse the parts of the spanning tree not affected by a failure of a given link, and replace the failed link of the tree with a different link originally not belonging to that tree, as shown in Fig. 10.

This idea was utilized, e.g., in [95], where a distributed mechanism was proposed based on the following three steps:

- *Failure detection*: detecting a failure in the physical layer;
- *Failure propagation*: broadcasting failure information in a "failure information propagation frame" which is assigned the highest priority;
- *Reconfiguration*.

The solution proposed in [95] not only avoids constructing and maintaining multiple spanning trees to protect against single link failures in two-connected networks, but it also creates new opportunities in terms of load balancing. Besides, it detects failures much faster than STP and RSTP, achieving recovery time values less than 50 ms, which is often faster than the other methods (see, e.g., Viking [93] described in Section III-B relying on a slower mechanism based on SNMP traps).

Another example of a spanning tree re-configuration strategy, called Fast Spanning Tree Reconnection (FSTR), is described in [98]. This distributed mechanism relies on an Integer Linear Program (ILP). It is executed offline to determine the best set of links that can be attached to the spanning tree to reconnect it after any single link failure. Whenever the failure occurs, switches incident to the failed link send notification messages containing the identifier of the failed link

to the preconfigured switches which can activate one of the available *reconnecting* links.

The related advantage is that only the directly affected switches and the intermediate switches need to update their tables, while the other devices do not have to. In this case, the recovery time consists mainly of the switch reconfiguration delay. The drawback is, however, that the existing Ethernet switches need to be modified to support the proposed solution. In particular, each compatible switch is required to maintain the following tables:

- *Failure notification table:* contains the MAC addresses of the switch interfaces where to send notification messages;
- *Alternate output port table:* includes the output port selected based on the identifier of the failed link (the table is computed offline).

After successful reconfiguration of switches, traffic forwarded via the failed link is instantly redirected, and the backward learning procedure is avoided. This mechanism also considers backup capacity guarantees.

As switches may not be aware of other failed links immediately, an improved distributed mechanism capable of dealing with double-link failures in Metro Ethernet Networks was proposed in [100]. The underlying concept remains similar to the authors' previous work from [98]. However, the main difference is that the reconnect links are selected in [100] in a way that loops are naturally avoided for double failures (even though each failure is handled independently).

The proposed ILP formulation includes an extension to determine the best set of reconnect-links that can reconnect each affected spanning tree while minimizing the backup capacity reserved in the network and satisfying the preferred protection grade for each connection. The new solution still deals with any single failure successfully. However, only partial protection can be provided for double failures.

The last concept we describe in this section is based on the idea of partial spatial protection (PSP) together with a mixed integer linear programming (MILP) is proposed in [103] as an extension of the two schemes ([98] and [100]) described above. In particular, as not all flows require full protection (against a failure of any possible link on a way), the extension described in [103] is to update the FSTR concept in a way to protect flows against a failure of a link from a specific subset of links only (to satisfy a given protection grade required by a demand).

B. Solutions Based on Multiple Spanning Trees

The **Multiple Spanning Tree Protocol (MSTP)** protocol is based on RSTP and it supports multiple spanning trees [81], [86], [111]. The main idea behind MSTP is to partition the network into multiple *regions* and calculate independent Multiple Spanning Tree Instances (MSTIs) within each of the regions based on the parameters conveyed in Bridge Protocol Data Unit (BPDU) messages which are exchanged between the involved network bridges.

The MSTIs are assigned unique Virtual LAN (VLAN) identifiers within the region so that frames with a given VLAN identifier are forwarded consistently by all bridges within that region. The consistent assignment is achieved based on MST Configuration Identifiers included in BPDUs that are transmitted and received by adjacent bridges in the same region. Such a mechanism is critical for a correct forwarding of frames within the region, as otherwise, some frames might be duplicated or even not delivered to the destination LANs. Note that no LAN can belong to two or more regions simultaneously.

Similarly to RSTP, MSTP defines a set of Port Roles which includes the *Alternate Port* and *Backup Port* roles, assigned to such ports of a bridge. They can be used to provide connectivity in the event of failure of other network components, or when other bridges, bridge ports, or LANs are removed from the network. In particular, an Alternate Port provides an alternate path to the one offered by the Root Port, in the direction of the Root Bridge. A Backup Port, however, can be used whenever the existing path offered by a Designated Port towards the leaves of the spanning tree becomes unavailable [111].

Initially, Alternate and Backup Ports are quickly transitioned to the Discarding Port state to avoid data loops. In the case of bridge or LAN failure, the fastest local recovery is possible when the Root Port can be substituted for the Alternate Port.

The related advantage is that as long as the Root Port Path Cost is equal for both ports, bridges located further from the Root Bridge will not see a network topology change [111]. However, in the other case, MSTP reconfigures the topology based on the spanning tree priority vectors, behaving like a distance-vector protocol. Note that during the reconfiguration phase, old information related to the prior Root Bridge may still circulate in the network until it ages out. For details related to the operation principles of MSTP, the reader is referred to [111].

A similar approach, which is also the main underlying concept of several other mechanisms, was proposed in Viking [93]. It was designed for a wide range of networking technologies, such as Local-Area Networks, Storage-Area Networks, Metropolitan-Area Networks, and Cluster networks, to provide fast recovery, high throughput, and load balancing over the entire network. The difference between Viking and MSTP is that Viking is based on multiple spanning trees covering the same network topology instead of covering particular disjoint segments of the network.

Viking relies on the VLAN technology widely supported in enterprise-grade Ethernet switches to control how packets are forwarded towards their destinations. In Viking, each packet carries a VLAN identifier associated with one of the available spanning trees. Based on that identifier, downstream switches forward the packet along the path in the corresponding spanning tree, as shown in Fig. 11. While relying on existing failure detection mechanisms implemented in modern network switches, Viking assumes that all end hosts run a local instance of the *Viking Node Controller* responsible for load measurements and VLAN selection. Whenever a link or node becomes unavailable, the centralized *Viking Manager* instructs the Viking Node Controllers using the affected VLANs to change the VLAN identifier carried in subsequent packets, effectively redirecting the related flows onto the precomputed alternative paths.

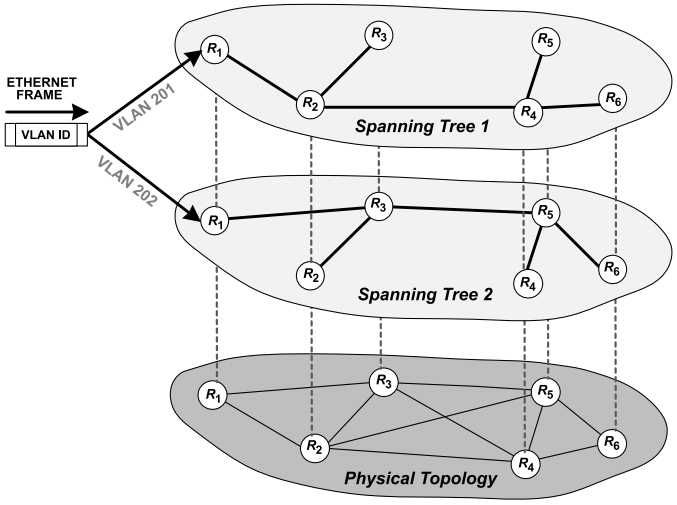

Fig. 11. Illustration of the spanning-tree switching process based on the VLAN identifier stored in each forwarded Ethernet frame.

Compared to the Ethernet architecture relying on a single spanning tree, the advantages of Viking include higher aggregate network throughput and the overall fault tolerance. Such an improvement is possible by reducing the downtime to sub-second values and preserving the existing higher-layer connections [93]. It operates in a semi-distributed way[1] and requires the SNMP protocol for internal signalling. The prototype developed by the authors relies on the Per-VLAN Spanning Tree (PVST) implementation by Cisco. As no firmware modifications are necessary, Viking can be deployed on many off-the-shelf Ethernet switches [93].

A distributed recovery scheme for a single-link failure scenario, which involves the "a priori" configuration of the alternative trees and the use of VLAN IDs to switch the traffic onto the alternative tree after a failure, was presented in [82]. In this scheme, the idea is to perform restoration locally by a node upstream to the failed element.

Two mechanisms of recovery are proposed, namely connection-based and repair-based recovery. In connection-based recovery, packets are assigned the backup VLAN ID based on the source node, destination node, and the primary VLAN ID. It means that traffic from different connections can be switched at a given node onto different backup spanning trees. At the same time, in the destination-based approach, packets are assigned the backup VLAN ID only based on the primary VLAN ID and the destination node. As a result, flows destined at a given node from different connections are switched at a given node onto the same backup spanning tree.

The latter approach is less complicated and involves a shorter computation time. However, as presented in the evaluation section of the paper, it is less capacity-efficient than the former scheme which determines the transmission paths based on a broader set of input parameters. To avoid forwarding loops, switching the traffic between spanning trees is allowed only once, and controlled by setting one bit in the Class of Service (CoS) field as the recovery bit.[2]

Another local protection scheme called EFUS based on the idea of multiple trees and the use of VLANs is presented in [84], [96]. Compared to the approach from [82], its advantage is the ability to provide recovery also in the case of failure of a single *node* if only the network topology is at least 2-connected. It is possible by the utilization of a pair of spanning trees for each network node, and by switching the flow onto the respective protection tree by the immediate upstream node onto the alternate spanning tree.

C. Solutions Based on Recovery Tables

In [99], a method to reduce service recovery time after a single link failure is proposed that is not based on spanning trees but uses the **concept of recovery tables** storing the alternate next-hop information for each entry in the conventional forwarding table. As proposed in [99], entries in the recovery table form the respective detours calculated based on information on the shortest paths maintained in the control plane provided by a routing protocol such as OSPF.

The scheme utilizes the concept of VLANs and changes the uppermost bit of the VLAN ID from 0 to 1 when redirecting the traffic based on the recovery forwarding table. Also, to avoid loops, it discards the packet if trying to redirect an already redirected one. However, the use of the uppermost bit of the VLAN ID for recovery indication limits the number of VLANs which can be established.

A resilience scheme for a spanning tree which is based on the idea of preconfigured protection paths, and tunnelling was introduced in [106]. To provide fast recovery after a link failure, protection paths are established before the occurrence of the failure using protection cycles (*p-cycles* originally proposed in [112] for ring-based networks in 1998) defined for each link in the spanning tree. After a failure, the node detecting the failure proceeds with the encapsulation of packets and forwarding them via the respective protection cycle to detour the failed link.

The advantage of this approach is a fast recovery of the affected flows being a major feature of the original *p*-cycles concept as well as resource efficiency (as *p*-cycles can provide protection not only for primary paths traversing the *p*-cycle but also those paths "straddling" the *p*-cycle).

D. Solutions Based on Message Flooding and Deduplication

As the performance of the link layer had already been identified as a growing problem, the AXE scheme was proposed in [104], [107] as a solution that not only retains the plug-and-play behaviour of the Ethernet but also ensures near-instantaneous recovery from failures and supports general network topologies. What distinguishes this fast-recovery mechanism from the widely-used Ethernet is that all network links can be used to forward packets, instead of using only a subset of links forming a spanning tree.

[1]Note that the authors planned to design and evaluate a fully distributed version in their second prototype.

[2]The CoS field is included in Ethernet frames under 802.1Q VLAN tagging [86].

In particular, AXE is based on the **flood-and-learn approach** and employs an AXE packet header containing the following four additional fields: the *learnable* flag `L`, the *flooded* flag `F`, a hopcount `HC`, and a nonce used by the deduplication algorithm. At the same time, AXE takes an orthogonal approach to the traditional flood-and-learn Layer-2 mechanisms such as STP in that it does not rely on any control plane at the link layer to compute the spanning tree. Instead, when an AXE node does not know how to reach a destination, it encodes a hop count in the packet header, sets the *learnable* flag in the packet header, and floods the packet throughout the entire network. While receiving multiple copies of the flooded packet, each node learns only the shortest path towards the destination. To this end, since flooding may reduce bandwidth resources, each node maintains a packet deduplication filter implemented as a cache, which avoids flooding packets in cycles — a catastrophic event in a network.

An advantage of AXE is undoubtedly also its compatibility with existing failure detection techniques, such as BFD and different hardware-based mechanisms. According to [104], even while using relatively small filters, AXE can scale to large Layer-2 topologies and can quickly react to failures by avoiding the STP-related computations.

E. Summary

In this section, we highlighted the mechanisms introduced in the literature to recover from failures of network elements at the link-layer level. Particular focus was on presenting the design characteristics contributing to the reduction of the recovery time. In this context, we first discussed the aspects of the conventional IEEE 802.1D Spanning Tree Protocol (STP), and in particular its remarkably slow convergence time measured even in terms of several tens of seconds. Next, we analyzed the representative schemes aimed at shortening the time of recovery grouped by us into four categories: (a) solutions based on a single spanning tree, (b) schemes utilizing multiple spanning trees, (c) techniques using the recovery tables, and (d) strategies based on message flooding and deduplication. In particular, schemes utilizing a single spanning tree such as, e.g., IEEE 802.1D Rapid Spanning Tree Protocol (RSTP) were found to require significantly less time to complete the recovery procedure of a spanning tree after a failure. By sending, e.g., explicit termination messages instead of using timeouts (as in [92]), or by executing the offline an Integer Linear Program to determine the best set of links for any single link failure scenario in advance ([98]), the recovery of a spanning tree could be completed by those methods only within several tens of milliseconds.

We then discussed the schemes utilizing multiple spanning trees, which can result in the partitioning of a network into subareas with one spanning tree installed in each such region and identified by a given VLAN ID (see, e.g., [111]). Such schemes were also shown in the respective literature to reduce the time of the recovery phase to sub-second values.

Schemes based on recovery tables (e.g., [99]) were next shown to be an essential alternative to tree-based techniques able to recover quickly from failures due to the pre-planned calculation of the alternate next-hops stored in recovery tables at each network node.

Finally, techniques based on message flooding and deduplication such as, e.g., AXE ([107]) relied on utilizing all network links (as opposed to schemes using only links belonging to spanning trees) to forward packets, as well as to recover quickly from failures.

Despite a broad set of link-layer fast-recovery mechanisms introduced in the literature, their limited implementation in practice remains an open issue. Indeed, apart from the standard solutions such as IEEE RSTP or MSTP, it is still rare to find the other fast-recovery mechanisms implemented in commercially available switches. Also, as link-layer recovery mechanisms were primarily designed for scenarios of single (link/node) failures, another open issue refers to their ability to recover from simultaneous failures of multiple network elements (i.e., due to an attack or another disaster-induced event).

IV. MPLS Fast Recovery

The architecture of Multiprotocol Label Switching (MPLS), introduced in [116], relies on Label Switching Routers (LSRs) capable of forwarding packets along Label Switched Paths (LSPs) based on additional labels carried in a packet header. Each of the MPLS labels assigns the packet to the corresponding Forwarding Equivalence Class (FEC) that defines a group of IP packets to be forwarded by one or more LSRs in a consistent manner. To be able to distribute information about the assignment of labels to the corresponding FECs among the LSRs in an automated way, a label distribution protocol may be deployed in the network. One of the available implementations is the Label Distribution Protocol (LDP) defined in [122].

As failures of network components are inevitable [139], LSPs might be disrupted from time to time. In that context, recovery strategies based on on-demand restoration (see Section II-B for the related discussion) may not always be an acceptable solution in terms of the recovery time scale, and thus, they are beyond the scope of this section. Instead, in the following parts of the section, we present and discuss different designs of fast-recovery mechanisms proposed for MPLS networks, focusing on those that redirect traffic to pre-established dedicated or shared backup LSPs quickly based on locally-available information. Additionally, the evaluation results are also reported for the selected solutions deployed in real test network environments, to provide valuable context related to their expected performance. Interested readers looking for structured information covering general terminology and schemes related to both protection and restoration techniques in the broader context of MPLS and Generalized MPLS (GMPLS) networks are referred to [73], [120], [121], [127].

A. MPLS Fast-Reroute Extensions

As MPLS had many advantages and was recognized to be a promising solution, it was soon extended to support additional much-needed functionalities, such as traffic

- **1999**
  - RFC 2702: Requirements for Traffic Engineering Over MPLS [114]
  - A Method for Setting an Alternative Label Switched Paths to Handle Fast Reroute (the first IETF draft; the last draft expired November 2000 [115])
- **2001**
  - RFC 3031: Multiprotocol Label Switching Architecture [116]
  - RSVP-TE: Extensions to RSVP for LSP Tunnels [117]
  - Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information [118]
  - Fast Rerouting Mechanism for a Protected Label Switched Path [119]
- **2002**
  - Fast Reroute Extensions to RSVP-TE for LSP Tunnels (the first IETF draft)
  - Multiprotocol Label Switching (MPLS) Traffic Engineering Management Information Base for Fast Reroute (the first IETF draft)
- **2005**
  - RFC 4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels [26]
- **2006**
  - RFC 4427: Recovery (Protection and Restoration) Terminology for Generalized Multi-Protocol Label Switching (GMPLS) [120]
  - RFC 4428: Analysis of Generalized Multi-Protocol Label Switching (GMPLS)-based Recovery Mechanisms (including Protection and Restoration) [121]
- **2007**
  - RFC 5036: LDP Specification [122]
- **2008**
  - RFC 5151: Inter-Domain MPLS and GMPLS Traffic Engineering – Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions [123]
  - Efficient Distributed Bandwidth Management for MPLS Fast Reroute [124] (based on earlier work published in 2005)
- **2010**
  - R3: Resilient Routing Reconfiguration [125]
  - Investigation of Fast Reroute Mechanisms in an Optical Testbed Environment [126]
- **2011**
  - RFC 6372: MPLS Transport Profile (MPLSTP) Survivability Framework [127]
  - Restoration Measurements on an IP/MPLS Backbone: The Effect of Fast Reroute on Link Failure [128]
  - RFC 6445: Multiprotocol Label Switching (MPLS) Traffic Engineering Management Information Base for Fast Reroute [129]
- **2013**
  - RFC 6981: A Framework for IP and MPLS Fast Reroute Using Not-Via Addresses [130]
  - Fast Reroute Based Network Resiliency Experimental Investigations [131]
  - Design Schemes for MPLS Fast ReRoute [132]
- **2014**
  - Fast ReRoute Model for Different Backup Schemes in MPLS Network [113]
- **2017**
  - RFC 8271: Updates to the Resource Reservation Protocol for Fast Reroute of Traffic Engineering GMPLS Label Switched Paths (LSPs) [133]
  - Fast ReRoute Scalable Solution with Protection Schemes of Network Elements [134]
- **2018**
  - Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks [135]
  - P-Rex: Fast Verification of MPLS Networks with Multiple Link Failures [136]
  - Linear Optimization Model of MPLS Traffic Engineering Fast ReRoute for Link, Node, and Bandwidth Protection [137]
- **2019**
  - RFC 8679: MPLS Egress Protection Framework [138]

[Year]

Fig. 12. Timeline of the selected documents and solutions related to MPLS Fast Reroute (entries marked in gray provide the general context related to the evolution of MPLS).

engineering [117] (further developed in the inter-domain context in [123]) and Fast-Reroute mechanisms to protect LSP tunnels [26].

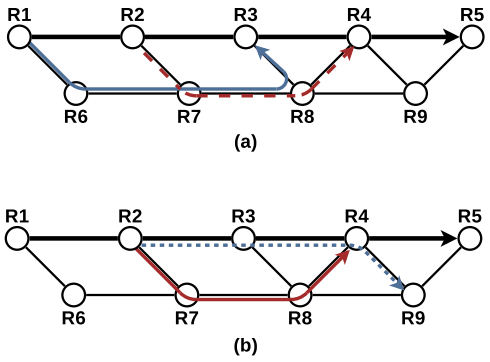

Fig. 13. Illustration of the two basic local protection methods defined in [26]: (a) *One-to-one backup* and (b) *Facility backup*. In the first figure, the thick black path leading from `R1` to `R5` represents the primary LSP, the thick blue path leading from `R1` to `R3` via `R6` — the backup LSP protecting node `R1`, and the thick dashed red path — the backup LSP protecting node `R2`. In the second figure, the thick solid black path and the blue dotted path represent the primary LSPs, while the thick red path leading from `R2` to `R4` via `R7` is the shared backup LSP which protects both primary LSPs if node `R3` or links between `R2-R4` fail.

The Fast-Reroute mechanisms in MPLS enabled the local repair of LSP tunnels within 10s of milliseconds based on the pre-established backup LSP tunnels. To meet this requirement, the following two local protection methods were defined (see Fig. 13 for the illustration of the underlying concepts):

- *One-to-one backup:* one backup LSP is established for each protected LSP in such a way that it intersects the primary path at one of the downstream nodes;
- *Facility backup:* one backup LSP is established to protect a set of primary LSPs in such a way that it intersects each of the primary paths at one of its downstream nodes.

In both cases, whenever a failure is detected by an LSR and the corresponding backup LSP is available, the LSR being the Point of Local Repair (PLR) performs a fast local failover, redirecting subsequent traffic to the preferred backup LSP. Based on the MPLS label stack carried in the message header, downstream LSRs will be able to recognize that incoming messages follow an alternative path, and they will forward the messages accordingly.

The two protection methods have one strong advantage: they are conceptually simple and can be deployed selectively where needed, together or alone, taking into account specific reliability requirements. Additionally, each of them is suitable for protection of links and nodes in the event of network failure. On the negative side, once deployed, they may not guarantee full failure coverage, even for single link failures.

The Resource Reservation Protocol — Traffic Engineering (RSVP-TE) Fast-Reroute procedures, originally defined in [26] to support the two methods discussed above, describe the necessary signaling mechanisms using the related RSVP objects that together enable fast-recovery capabilities in RSVP-TE MPLS networks. The primary specification has recently been updated in [133] to support Packet Switch Capable GMPLS LSPs. In particular, new signaling procedures and a new `BYPASS_ASSIGNMENT` subobject in the RSVP `RECORD_ROUTE` object are defined, to coordinate the assignment of bidirectional bypass tunnels

which protect common facilities in both directions along the corresponding co-routed bidirectional LSPs.

The proposed modifications have several advantages. First, bidirectional traffic can be redirected onto bypass tunnels in such a way that the resulting data paths are co-routed in both directions. Second, the proposed fast-reroute strategy may either be used with GMPLS in-band signaling or with GMPLS out-of-band signaling, offering greater flexibility during the potential deployment. Finally, it is also possible to avoid the RSVP soft-state timeout in the control plane which would typically occur after one of the downstream nodes stopped receiving the `RESV` messages associated with the protected LSP from the upstream PLR — for example, as a result of unidirectional link failures. For more detailed technical information about the related RSVP-TE extensions and objects, the reader is referred to [26], [133].

Protection of egress links and egress nodes: While the local-repair-based MPLS Fast-Reroute mechanisms discussed above mainly focus on transit link/node protection, the MPLS Egress Protection Framework recently introduced in [138] describes fast-recovery mechanisms designed to protect egress links and egress nodes in MPLS networks relying on downstream-assigned service labels. In principle, the proposed solution relies on bypass tunnels pre-established by the PLR to one or more predefined nodes called *protectors*. A protector may either be physically co-located with or decoupled from a backup egress node, and in the case of failure of the egress node, the protector is expected to forward the service packets rerouted at the PLR through the bypass tunnel towards the backup egress node. It is assumed that multiple egress nodes are available in the MPLS domain, and that for each protected tunnel, one router is designated to be the protector. Further, to enable fast recovery, the necessary bypass forwarding state needs to be pre-installed in the data plane. Each of the protectors is also expected to perform context label switching and context IP forwarding for rerouted MPLS and IP service packets, respectively.

One strong advantage of the proposed framework is that it supports networks with co-existing tunnels and services of different types. It may also complement existing global recovery schemes as well as control-plane reconvergence mechanisms. On the negative side, the current specification does not support services using upstream-assigned service labels.

Fast ReRoute using Not-via Addresses (Not-via) is a concept based on a single-level encapsulation and forwarding of packets to specifically reserved IP addresses which are also advertised by the IGP. In this way, it is possible to protect unicast, multicast, and LDP traffic against failure of a link, a router, and a shared-risk group. Note that we already provide a detailed discussion of this mechanism in the context of IP networks in Section V-B. Interested readers may also study [130] where the general approach to using Not-Via Fast-Reroute in MPLS networks is discussed.

B. Methods Based on Optimization

An important subgroup of MPLS Fast-Reroute solutions relies on mathematical optimization to improve the overall performance with respect to different factors, such as: resource utilization, coverage of failure scenarios, and scalability. We summarize the selected approaches below.

Distributed bandwidth management and backup path selection: To be able to perform the failover from the primary LSP to a precomputed backup LSP in a given failure scenario, the network requires additional resources, such as available bandwidth on network links along the related LSPs. On the other hand, the resources that remain reserved specifically for the purpose of the possible recovery cannot be assigned to other primary LSPs, and thus should be minimized. However, they can be shared by two or more backup LSPs, as long as the corresponding primary LSPs do not fail simultaneously. An illustration of this approach in the context of distributed bandwidth management mechanisms supporting MPLS Fast Reroute is provided in [124]. In principle, the authors focus on the one-to-one protection method and single link or node failures. First, to collect the necessary information about the reserved and available bandwidth as well as the administrative link weights, traffic engineering extensions to the selected link-state routing protocol might be used.[3] Then, the primary LSPs are computed using the constrained SPF algorithm based on the administrative weights as well as the available bandwidth on the involved unidirectional links. To be able to reserve the necessary amount of resources on network links while a new primary LSP is created, the RSVP-TE extensions are used, which involves the exchange of the `PATH` and `RESV` signaling messages between the source and destination nodes. In particular, the `RESV` message is sent from the destination to the source. Whenever it is received by an intermediate node along the primary LSP, the node executes the constrained SPF algorithm to find a suitable local backup LSP towards the destination, excluding the downstream link belonging to the primary LSP. Subsequently, once the backup LSP has been selected by the intermediate node, the `PATH` message is sent along the backup path to establish the LSP and reserve the required amount of shared restoration bandwidth. Meanwhile, each of the downstream nodes maintains a local array for each unidirectional link starting at that node, to keep track of the required restoration capacity on the related unidirectional link in different failure scenarios. The `PATH` message allows the downstream nodes to update the current values in the respective local arrays based on the following information included in the message:

- the immediate downstream link on the primary path;
- the immediate downstream node on the primary path;
- the requested bandwidth of the primary LSP.

Once the local arrays have been updated accordingly, the new estimated values of the available and reserved bandwidth on the involved unidirectional links will be disseminated to the other nodes by the extended link-state routing protocol in use. As soon as the primary LSP is no longer needed, the `PATHTEAR` signaling message can be sent along the primary LSP to tear the LSP down. Note that each intermediate node will also need to send the `PATHTEAR` message along the

[3]Note that OSPF and IS-IS are example link-state protocols that have been extended accordingly.

local backup LSPs established earlier, to release the reserved network resources properly. In this specific case, as the local arrays must be updated accordingly, it is proposed that the PATHTEAR message include additional information about immediate downstream link and node belonging to the primary LSP. As an additional improvement, the authors present and discuss a way to optimize the backup path selection procedure, aiming at reducing the overall required restoration bandwidth. Although the additional information required by the proposed optimized backup path selection algorithm is distributed between adjacent routers using three new signaling messages (TELL, ASK, and REPLY), the authors suggest that some information be embedded in the existing PATH and RESV messages. In such a case, only one new signaling message would need to be introduced in real deployments. One strong advantage of the proposed solutions is that they do not rely on centralized mechanisms. On the negative side, the involved LSRs would need to support the required signaling extensions.

Resilient Routing Reconfiguration (R3) is a concept proposed to address the long-standing shortcomings of the earlier fast-recovery techniques with respect to insufficient performance predictability and missing protection mechanisms against possible network congestion, especially in multi-failure scenarios [125]. R3 is based on the following two main steps:

1) *Offline precomputation*: find a suitable routing scheme and protection (rerouting) scheme for a given traffic matrix, to minimize the maximum link utilization over the entire network;
2) *Online reconfiguration*: in the case of failure, reroute traffic via alternative paths and adjust the routing and protection schemes to exclude the failed link from the set of candidate links in the event of subsequent failures.

The first step is not time critical and relies on linear programming duality to convert the related primary optimization problem with an infinite number of constraints into a simpler form containing a polynomial number of constraints.[4] Although the corresponding model contains $O(|V|^4)$ variables and $O(|V|^4)$ constraints, where V denotes the set of nodes in the network graph, the authors emphasize that this is much lower than for the other existing approaches focused on oblivious routing [141], [142].

In the second step, it is important that the rerouting decision be made as soon as possible, to avoid packet losses and increased delay. Thus, the related operations are designed not to be computationally intensive. Additionally, the routing and protection schemes at all involved routers may be updated after the upstream PLR has started rerouting packets via the selected detour, without affecting the recovery process.

One strong advantage of R3 is that it can deal with both multiple link failures and traffic variability. Additionally, beyond performing extensive simulations, the authors have also implemented and evaluated R3 in a real testbed, which confirmed its effectiveness. On the negative side, the implementation prepared by the authors relies on a centralized precomputation of the protection routing scheme. At the same time, the offline precomputation task does not have a direct influence on the duration of the recovery phase.

Primary and backup path computation is a typical problem in the context of MPLS Fast Reroute. Beyond the solutions presented above, it was also discussed in [113], [132] where the authors proposed the corresponding non-linear integer programming models related to link, node, or path protection schemes. To reduce the expected computational complexity, subsequent efforts were made to prepare the linear variants of the related optimization problems which also take into account the available resources on network links [134], [137].

C. Fast Verification of MPLS Networks

Testing and debugging data plane configurations is generally considered a difficult manual task, yet, a correct configuration of the data plane is mission critical to provide the required properties in terms of policy compliance and performance. Reasoning about the data plane behavior *subject to failures* is particularly challenging, as it seemingly introduces a combinatorial problem: it seems unavoidable that one has to test each possible failure scenario individually, and simulate the resulting rerouting, in order to verify that the network behaves correctly under all failure scenarios.

Interestingly, this intuition is wrong: it has recently been shown that in the context of MPLS networks, it is possible to conduct *what-if analyses* in *polynomial time*. In [135], an approach is presented to collect and transform MPLS forwarding and failover tables into a prefix rewriting system formalism, to which automata-theoretical algorithms can be applied to test a wide range of properties, related to reachability (e.g., can *A* always reach *B*, even if there are up to 3 link failures?) or network policy (e.g., is it ensured that traffic from *A* to *B* traverses a firewall along the way?).

The fast verification is enabled by the nature how labels are organized and manipulated in the MPLS packet header: the labels are organized in a *stack*, and operations limited to *push*, *pop*, and *swap*. This makes it possible to describe the system as a push-down automaton. In [136] a tool called *P-Rex* is presented which realizes the theoretical concepts in [135]. *AalWiNes* [143] extends P-Rex in that it relies on weighted automata, allowing to verify quantitative properties as well (besides being faster).

To the best of our knowledge, no polynomial-time solutions exist for conducting what-if analyses in polynomial time for other types of networks which rely on more complex rules, e.g., [144], [145].

D. Other Approaches

Beyond the groups of solutions summarized in the sections above, each of them sharing a common design feature, there are also other fast-recovery concepts related to MPLS which are based on interesting ideas, and thus are also worth mentioning. We discuss the selected proposals in this section.

Aggregated Link Usage Information: The idea of sharing some backup resources in the network by multiple backup bandwidth-guaranteed LSPs has already been considered in [118] where the authors discuss the trade-off

[4]Interested readers may learn the basics of optimization theory from [140].

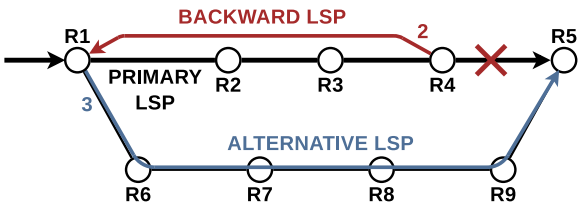

Fig. 14. Illustration of the underlying idea of the improved protection methods described in [119], [146].

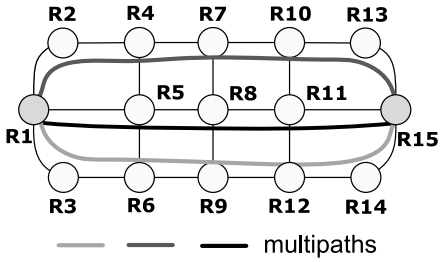

Fig. 15. Example configuration of a multipath.

between complete knowledge of the routing scheme at the time of a path setup event and partial knowledge based on the aggregated link utilization information. Maintenance of non-aggregated per-path information may cause potential scalability issues and requires more storage and processing resources. On the other hand, the solution proposed by the authors relies on such information as fraction of bandwidth used on each link by the active LSPs and separately by all backup LSPs. Based on that, the proposed routing algorithms are able to determine the candidate backup paths for local recovery to protect single link or node failures.

Using Backward LSPs: A fast-recovery method based on diverting traffic back towards the upstream LSR and selecting one of the predefined alternative LSPs was proposed in [115] and improved further in [119], [146]. The underlying idea of the related fast-recovery strategy is illustrated in Fig. 14.

Once a failure is detected by one of the LSRs on the primary LSP, packets are sent along the *backward* LSP towards the upstream LSR. The upstream LSR recognizes the backward flow, marks the last packet sent along the broken LSP using one bit of the `Exp` field of the MPLS label stack (note that no overhead is introduced at this point), and stores the subsequent packets received from the upstream LSR in a local buffer to avoid packet reordering. As soon as the previously marked packet is received again from the downstream LSR, all related packets stored in the buffer are forwarded to the upstream LSR. Eventually, the source LSR of the protected LSP redirects packets onto the predefined *alternative* LSP.

While the original method already allowed for a significant reduction of the overall path computation complexity and signaling requirements,[5] the improved method was designed to eliminate packet reordering which was one of the disadvantages of the earlier proposals. The average delay during the restoration period has also been improved. It is worth noting that this recovery strategy may be used to achieve one-to-one and many-to-one protection faster as well as to avoid network congestion, allowing for better QoS control and reduced packet losses [119], [146]. More importantly, by introducing relatively small packet buffers at LSRs to be able to store copies of the limited number of forwarded packets, it is possible to eliminate packet losses entirely and thus improve the observed TCP performance during the failover considerably [146]. At the same time, the proposed two improved approaches do not specify any particular method for the effective selection of alternative LSPs — an important factor having significant influence on the QoS and the observed path stretch. Interestingly, the authors of [115] already anticipated the potential problems in the context of delay-sensitive network services and outlined the concept of *restoration shortcuts* as one of the possible ways to counteract the expected increase of transmission delay as a result of the failover. In particular, while using a restoration shortcut, traffic is rerouted over an alternative *shortcut* LSP established between the LSR on the primary path (upstream of the failed link) and the destination of the primary LSP, potentially merging into the other existing backup LSPs.

Leveraging the Multipath Transmission: Resilience in MPLS networks can also be provided by multipath structures being a type of protection-switching mechanisms. One of the representative schemes of this kind is the self-protecting multipath (SPM) concept proposed for MPLS networks in [147].

As presented in Fig. 15, for a given end-to-end demand d, in the normal operation mode, SPM uses all pre-established paths for data transmission. The multipath structure should include paths being mutually node-disjoint, to ensure protection against a failure of a single node/link as well as to simplify the procedure to establish the multipath. Load balancing among several paths of the multipath is also helpful while dealing with failures. In the case of any failure affecting a given path in the multipath structure, traffic is redistributed across all the other working paths. Therefore, for a multipath consisting of k paths, SPM needs $k + 1$ different traffic distribution functions: one for operation in the failure-free scenario, and the next k functions to cover the failure of any of the k paths.

The evaluation presented in [147] shows that the approach is very efficient in protecting against failures of single nodes and links, as it requires only about 20% of additional transmission capacity for this purpose.

One of the variants of SPM has been presented in [148], with the extension to introduce the traffic distribution function specific to the failed network element along the path, instead of the former traffic distribution function defined in the context of a given path.

The SPM concept has been enhanced in [149] with the proposal of a linear program (LP) to optimize the SPM load balancing function to maximize the amount of traffic transported with resilience requirements for legacy networks (i.e., already deployed networks). The objective is achieved by

[5]In addition, note that the computations related to primary and alternative paths may be performed at a single switch, to avoid possible issues resulting from distributed computation.

solving the corresponding problem of minimization of the maximum link utilization in any protected failure scenario.

E. Experimental Evaluation

Some of the existing solutions have been evaluated in real test network environments. As the corresponding results may provide a valuable context with respect to the expected performance, we summarize below the selected reports related to fast-recovery mechanisms.

The first paper discussed in this section presents the comparison of the following two mechanisms with respect to packet losses and duration of the failover phase [126]:

- MPLS Traffic Engineering (TE) Fast Reroute [26], [114], [129];
- IP Fast Reroute Framework [150].

The evaluation scenarios assumed single link failures and protection based on either predefined backup paths using MPLS TE Fast-Reroute tunnels or IP Fast-Reroute Loop-Free Alternates (for a detailed discussion of IP Fast-Reroute solutions, the reader is referred to Section V). The results presented by the authors suggest that with increasing number of rerouted LSPs in MPLS networks, the failover time increases exponentially (even beyond 50 ms) and packet losses are also higher. In the case of the investigated IP network, increasing the number of IP prefixes also causes a non-linear increase of the failover time, while traffic losses remain almost unchanged.

In contrary to the experiments reported in [126], the second analyzed paper presents the results from a large and geographically-distributed production backbone network in which every major routing node consisted of several core, aggregation, and edge routers [128]. The observed parameters included TTL changes, packet losses, packet reordering, and one-way delay changes. The experiments were performed over a 14-month period and the two considered restoration methods (OSPF reconvergence and MPLS-TE Fast Reroute) were analyzed during seven consecutive months each. The evaluation results have shown that the MPLS-TE Fast-Reroute mechanism reduces packet losses and packet reordering during link failure events significantly, also suppressing the possible micro-loops.

The third considered paper compares the effectiveness of MPLS-TE Fast Reroute and IP OSPF Fast Reroute based on Loop-Free Alternates in terms of packet losses and network convergence time, for different numbers of LSPs and IP prefixes [131]. Both mechanisms were configured in a WDM test network. The reported results confirm that with increasing number of rerouted LSPs in MPLS networks, the convergence time increases significantly and the observed packet losses are also higher. Again, in the case of the investigated IP network, increasing number of IP prefixes caused a non-linear increase of the convergence time, while traffic losses remained almost unchanged.

Fast-recovery mechanisms designed for MPLS have been shown to improve network operation and performance in different failure scenarios. At the same time, there are still some related open challenges. In particular, fast restoration relying on backup LSPs requires that additional LSPs be configured and established in advance and in an effective way, taking into account the trade-off between the coverage of failure scenarios, the available network resources, expected traffic demands, key performance indicators from the perspective of users, and the overall complexity of the system. Moreover, considering the variety of failure detection and mitigation mechanisms running concurrently on different layers of networked systems, another challenge is how to avoid packet losses and possible packet reordering during the failover.

V. Intra-Domain Network-Layer Fast Recovery

In this and the next sections, we review the representative schemes for fast recovery that operate at the network layer (Layer 3) of the Internet protocol stack. In particular, this section is dedicated to the intra-domain setting, while the next section will discuss the solutions that work across Autonomous Systems (inter-domain). Since the prevailing network-layer protocol today is the Internet Protocol suite, namely the two versions IPv4 and IPv6, most of the fast recovery schemes we discuss here are specifically designed for IP (IP Fast ReRoute, see below). However, the main ideas will be reusable in any network layer that provides connectionless, unreliable datagram service [151].

In the context of a single provider network, the aim of the operator is to achieve the highest possible level of failure resilience and the lowest service recovery time. As outlined in Section II, in general this is attained by *preplanned* protection schemes (with backup paths established before the failure), providing detours over small parts of working paths (i.e., *local detours*) with backup capacity reserved in advance to ensure an undisrupted flow of traffic after a failure (*dedicated* protection). For standard IP networking, however, it is not common to see these objectives fulfilled jointly due to the connectionless nature of the IP network layer that does not allow packets to be "pinned" to a preplanned detour after a failure. Therefore, IP Fast-Reroute (IPFRR) mechanisms need to tediously work around the limitations of the underlying network layer, and typically rely on shared detours and provide "best-effort" protection only. In other words, in IP failure recovery, usually there is no guarantee that the necessary capacity and network resources remain available along the backup paths after a failure event (see, e.g., [152], [153]).

Below, we review the most important intra-domain fast IP recovery schemes following a rough chronological order and we provide a simple taxonomy for classifying the schemes. We note, however, that our coverage of IP Fast ReRoute and related concepts is intentionally incomprehensive. In particular, we deliberately ignore control-plane driven IP restoration schemes (which tend to be slower) and we concentrate on very fast data-plane driven *shared, preplanned, local protection schemes* exclusively (recall Fig. 6), which either work on top of the unmodified connectionless destination-based unicast IP data plane service and a distributed intra-domain IP control-plane protocol or require only minimal extensions to the bare-bone IP specification [154]–[156]. For fast IP restoration, see [15], [157], [158], for pointers on IP multicast fast recovery see, e.g., [159]; and for schemes that do not fit into

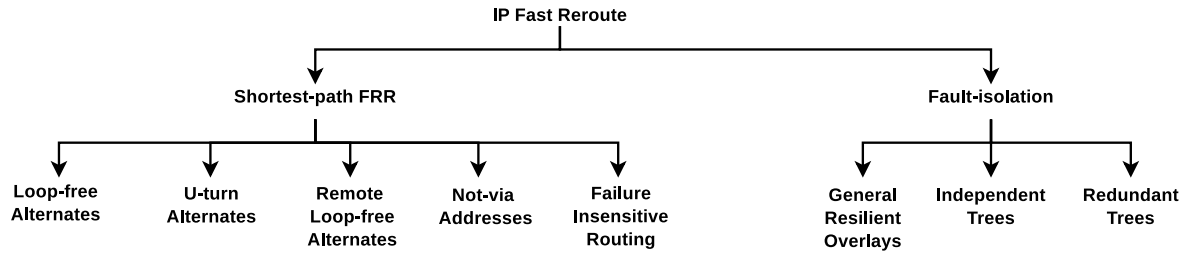

Fig. 16. A taxonomy of the most important general concepts in IP Fast ReRoute.

this pure IP data-plane "ideal", like O2 routing [160], Failure-carrying Packets (FCP) [161], Protection routing [152], [153], or KeepForwarding [162], see Section VII. We largely ignore the intricate issues related to IP fault detection and fault localization, noting that IP recovery usually relies on (from the slowest to the fastest technique) control plane heartbeats, Bidirectional Forwarding Detection, and link-layer notification, see Section II-C. For further detail on IPFRR refer to [23], [24], for the algorithmic aspects see [163]–[167], and for a comprehensive evaluation and comparison of different IPFRR techniques see [35], [168]. For a taxonomy of the most important concepts in IPFRR, see Fig. 16, and a timeline of related research and standards, see Fig. 17.

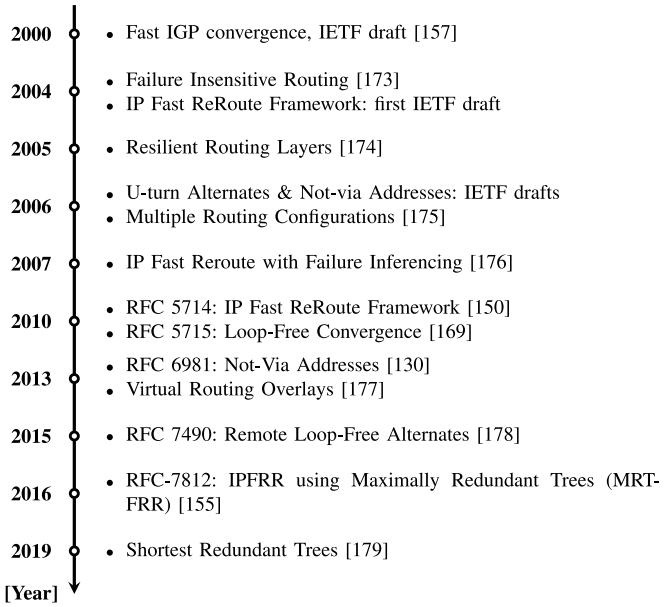

Fig. 17. Timeline of selected documents and solutions related to IP Fast Reroute.

A. The IP Fast ReRoute Framework

When a link or node failure occurs in a routed IP network, there is inevitably a period of disruption to the delivery of traffic until the network re-converges on the new topology. Packets for destinations that were previously reached by traversing the failed component may be dropped or may suffer looping [169]–[171]. Recovery from such failures may take multiple seconds, or even minutes in certain cases, due to the distributed nature of the IP control plane and the complex interactions between intra-domain Interior Gateway Protocols (IGPs) and inter-domain exterior gateway protocols [15], [157], [158]. Many Internet applications (multimedia, VPN) have not been designed to tolerate such long disruptions.

In this section, we review the *IP Fast ReRoute framework* (IPFRR [150]), a mechanism to provide fast data-plane-driven failure mitigation in an intra-domain unicast setting. The first specification for IPFRR was drafted in 2004, reaching the status of an Informational RFC to 2010 [150].

Reports suggest that roughly 70% of outages in operational IP backbones are *local*, affecting only a single link and only a single Autonomous System (AS) at a time, and *transient*, lasting only a couple of seconds; think of, e.g., a flapping interface or a router in a quick reboot cycle [139], [158], [172]. The main premise of IPFRR is that for such transient local failures, it is an overkill to execute two full AS-wide IGP re-convergence processes just to eventually return to the original configuration, since IGPs are usually too slow for operational purposes [15], [157], [158].

The main goal of IPFRR is (1) to handle short-term disruptions efficiently and (2) remain fully compatible with the IP control plane and data plane, allowing for the incremental deployment with no flagship date. The framework rests on two main design principles: *local rerouting* and *precomputed detours* (recall the general taxonomy in Section II-B). Local rerouting means that only routers directly adjacent to a failure are aware of it, which eliminates the most time-consuming steps of IGP-based restoration, the global flooding of failure information. Additionally, IPFRR mechanisms are proactive in that detours are computed, and installed in the data plane, before a failure occurs. Thus, when a failure eventually shows up, the affected routers can switch to an alternate path instantly, letting the IGP to converge in the background.

The IPFRR framework distinguishes *local repair paths* (ECMP and LFA, see Section V-B), the cases when a router has an immediate IP-level neighbor with a path that is still functional after the failure, and *multi-hop repair paths*, whenever the closest router with a functional repair path is multiple IP links away and, therefore, is not available directly via a local interface (rLFA).

B. Shortest-Path Fast Reroute

Most intra-domain IP routing protocols, like OSPF (Open Shortest Path First [10]) or IS-IS (Intermediate System-to-Intermediate System [11]), rely on a flooding mechanism to distribute network topology information across the routers inside an AS and a shortest-path-first algorithm, i.e., Dijsktra's

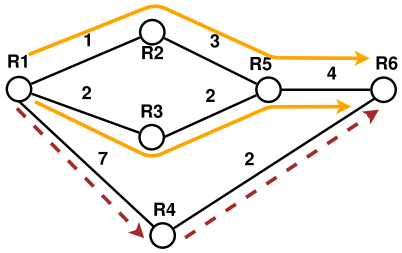

Fig. 18. Illustration of Loop-free Alternates and ECMP alternates with link costs as marked on the edges. For source router R1 towards destination router R6, both R2 and R3 are ECMP next-hops (marked by orange arrows in the figure), each one providing a node-protecting ECMP LFA with respect to the case if the other one fails, and router R4 is a link- and node-protecting LFA protecting against the (potentially simultaneous) failure of R2 and/or R3 (the backup route is marked by a dashed red arrow in the figure).

algorithm [32], to calculate the best route, and the primary next-hop router(s) along these routes, to be loaded into the data plane. The idea in all LFA-extensions discussed in this section is to *leverage the IGP link-state database*, readily available and synchronized across all routers, to compute not only a primary next-hop to each destination, but also *to obtain one or more secondary next-hops* as well that can be used as bypass whenever the failure of the primary next-hop is detected. As such, shortest-path-based IPFRR is generally easy to implement and incrementally deploy in an operational network. This, however, comes at the price of a major limitation: namely, as *the bypass paths themselves must also be (at least partially) shortest paths*, a proper shortest bypass path may not always be available in a given topology for a particular failure case. Correspondingly, in general it is very difficult to achieve 100% protection against all possible failure cases with shortest-path-based IPFRR methods.

Loop-free Alternates (LFA) is the basic specification for the IP Fast-ReRoute framework to provide *single-hop repair paths*. In LFA, the emphasis is on simplicity and deployability, instead of full coverage against all transient failures [32], [180]. Drafted in the IETF Routing Working Group in 2004, LFA reached Standards Track RFC status in 2008 (two years before the actual IPFRR framework specification was finalized [150], see Fig. 17).

As mentioned above, the idea in LFA is to exploit the information readily available in the IGP link-state database to compute secondary next-hops, or "alternates" as per [32], that can be used as a bypass whenever the failure of the primary next-hop is detected. Computing these secondary next-hops must occur in a "loop-free" manner so that the bypass neighbour, which, recall, will not be explicitly notified about the failure event, will not loop the packets back to the originating router. LFA uses some basic graph-theory and simple conditions based on the shortest-path distances calculated by the IGP to ensure that the calculated alternate routes are indeed loop-free.

The conditions based on which routers can choose "safe" alternate next-hops are as follows. Given router s, destination prefix/router d, let e be a shortest-path next-hop of s towards d (there can be more than one next-hop, see below). In addition, let dist(i, j) denote the shortest-path distance between router i and j. Then, from s to d with respect to the next-hop e, a neighbour $n \neq e$ of s is

- a *link-protecting LFA* if

$$\text{dist}(n, d) < \text{dist}(n, s) + \text{dist}(s, d), \tag{2}$$

- a *node-protecting LFA* if, in addition to (2),

$$\text{dist}(n, d) < \text{dist}(n, e) + \text{dist}(e, d), \tag{3}$$

- a *downstream neighbour LFA* if

$$\text{dist}(n, d) < \text{dist}(s, d), \tag{4}$$

- and an ECMP alternate if

$$\text{dist}(s, n) + \text{dist}(n, d) = \text{dist}(s, d). \tag{5}$$

The definitions follow each other in the order of "strength" and generality: for instance, an ECMP alternate is always a downstream neighbour, and a node-protecting LFA is also link-protecting. For the exact relations among different LFA types, see [32], [180], and for an illustration of the main concepts in LFA, refer to Fig. 18. In general, a "stronger" notion of LFA alternate next-hop should always be preferred over a "weaker" one whenever multiple choices are available. An algorithm to choose the best option in such cases is specified in [32].

The main advantages of LFA are that it is simple to implement and fully-compatible with the deployed IP infrastructure. Correspondingly, LFA is readily available in most major router products [181]–[183]. Nevertheless, LFA comes with a number of distinct disadvantages as well. First, it needs an additional run of Dijkstra's algorithm from the perspective of each LFA candidate neighbour to obtain the shortest path distances, causing the extra control CPU load. Second, LFA does not guarantee full failure coverage: depending on the topology and link costs LFA can protect about 80% of single link failures and 40–50% of node failures in general. Correspondingly, various optimization methods are available in the literature to improve failure-case coverage in operational networks [184]–[189]. Third, LFA is prone to forming transient "micro-loops" during recovery, caused by certain routers still using the "normal" routes while others already switching to the recovery routes [32], [190], and possibly "LFA loops" as well that may show up after all routers have switched. For instance, using a link-protecting LFA to protect a node failure may generate an LFA loop that will persist until the IGP finishes full reconvergence in the background [32], [191], [192]. In general, there is a trade-off between loop-avoidance and failure-coverage. For example, using only the strong notion of a downstream neighbour (see Eq. (4)) as an LFA eliminates both micro- and persistent LFA loops, but the failure case coverage attained this way may be poor in certain provider networks [180].

U-turn Alternates is an extension to the basic LFA specification providing multi-hop repair paths, on top of the local-repair mechanism implemented by LFA, in order to improve the failure case coverage [193]. The U-turn alternates specification never reached RFC status.

The main observation in the U-turn alternates specification is that the only way for a router to remain without an LFA is

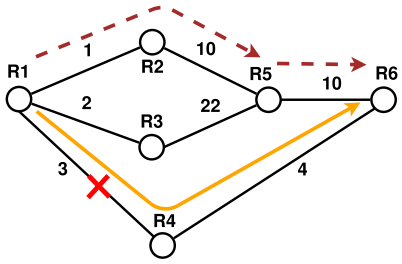

Fig. 19. Illustration of U-turn Alternates and Remote Loop-free Alternates with link costs as marked on the edges. Source router R1 has router R4 as its default shortest-path next-hop towards router R6 (marked by an orange array in the figure) and there is no available LFA candidate neighbour that would provide protection against the failure of the link R1–R4 (marked by a red cross), or the next-hop router R4 itself. However, router R5 is both a U-turn alternate and a Remote Loop-free Alternate in this case: note that both R2 and R3 are possible U-turn alternate next-hops that can pass the packet to R5, but in the case of rLFA the bypass path will be provided along the shortest R1–R5 route exclusively (i.e., by router R2), yielding the new path R1–R2–R5–R6 (marked by dashed red arrays).

if it is itself one of the default next-hops of each of its neighbours. Then, a U-turn alternate is a neighbour that has a further neighbour (at most two hops away) that still has a functional path to the destination after the failure. Consequently, that "next-next-hop" neighbour of the U-turn alternate can be used as a bypass whenever the primary next-hop fails and no LFA is available. Unfortunately, this requires some external mechanism to prevent the U-turn alternate to loop the packet back to the originating router (which would be the "default" behaviour). See Fig. 19 for an illustration.

Technically, for router *s*, destination prefix/router *d*, a neighbour *n* of *s* is a *U-turn alternate* next-hop from *s* to *d* if (1) *s* is the primary next-hop from *n* to *d* and (2) *n* has a node-protecting loop-free LFA to *d* by (3). In such cases, *s* can send a packet to *n*, which, detecting that a packet was received from its primary next-hop (this can be detected by, e.g., Reverse Path Filters [194]), can send it along its node-protecting LFA. Alternatively, a packet sent to a U-turn alternate can be explicitly marked to signal that it should not be looped back to the originating router.

The U-turn alternates specification is relatively simple to implement. On the other hand, it needs RPF, per-packet signalling, tunnelling, or interface-specific forwarding (see below) to indicate that packet is travelling to a U-turn alternate, plus an additional run of Dijkstra's algorithm from each neighbour of every U-turn candidate. Still, depending on the topology U-turn alternates cannot guarantee full failure coverage, not even for single link failures. To account for non-complete coverage, [195] presents an extension where multiple U-turns may come one after the other. This provides 100% protection at the cost of worsening the issues related to the signalling of U-turns.

Remote Loop-free Alternates (rLFA) is another extension of the basic LFA specification to extend the scope to multi-hop repair paths [166], [167], [178], [196]. Again, the intention is to improve LFA failure case coverage.

The most important observation that underlies rLFA is that any remote router may serve as an alternate, provided that (1) the remote router has a functional path to the destination after a failure, (2) the shortest path from the originating router to the alternate does not traverse the failed component, and (3) the originating router has some way to tunnel packets from itself to the alternate. In such cases, whenever a router needs to find a bypass to divert traffic away from a failed primary next-hop, it can encapsulate these packets to tunnel the diverted traffic to the remote loop-free alternate (using, e.g., IP-IP, GRE, or MPLS), so that the router at the tunnel endpoint will pop the tunnel header from the packets and use its default next-hop to reach the destination. See Fig. 19 for a sketch of the main ideas in rLFA.

In order to check whether a remote router is a valid rLFA candidate, the shortest path segment from the originating router to the rLFA and from the rLFA to the destination router both must avoid the failed component. The second condition is compatible with the LFA loop-free condition, whereas the first condition is needed because the encapsulated packets, travelling from the originating router to the rLFA, will also follow the shortest path and, as such, may also be affected by the failure.

Formally, for router *s*, destination router/prefix *d*, and next-hop *e* from *s* to *d*, some $n \in V$ (not necessarily an immediate neighbor of *s*) is a *link-protecting remote loop-free alternate* (rLFA) if the below two conditions hold:

$$\mathrm{dist}(s,n) < \mathrm{dist}(s,e) + \mathrm{dist}(e,n) \quad \text{and} \tag{6}$$

$$\mathrm{dist}(n,d) < \mathrm{dist}(n,s) + \mathrm{dist}(s,d). \tag{7}$$

The node-protecting case can be defined similarly [166].

The pros of rLFA are that it remains largely compatible with IP and it is also straight-forward to implement on top of MPLS/LDP and segment routing [197].

On the negative side, rLFA may cause extra control CPU load as it needs each router to execute a shortest-path computation from potentially each other router, and extra data-plane burden by routers having to maintain possibly a huge number of tunnels to reach each rLFA. Crucially, rLFA may still not provide full protection, not even in the unit-cost case. Papers [166], [167] provide analytical and algorithmic tools to characterize rLFA coverage in general topologies. To address the inadequacy of the failure case coverage, the originating router may use *directed forwarding* to instruct the rLFA to send a packet through a specific neighbour. This modification guarantees full coverage for single-link failures [196], [198]. Interestingly, this use case served as one of the precursors for the development of segment routing [197] and the main motivation to develop a family of LFA extensions that provide complete failure case coverage in the context of this emerging segment routing framework [199]. We note that the rLFA specification largely replaced U-turn alternates.

IP Fast ReRoute using Not-via Addresses (Not-via) is a specification for fast IP failure protection that addresses the limitations of LFA and rLFA. Drafted in 2005, the specification reached RFC status in 2013 [130] but major vendor adoption and large-scale deployments did not ensue.

The main drawback of rLFA is that even if a suitable remote LFA candidate is available after a failure the originating router may not have a way to send bypass packets there, since all its

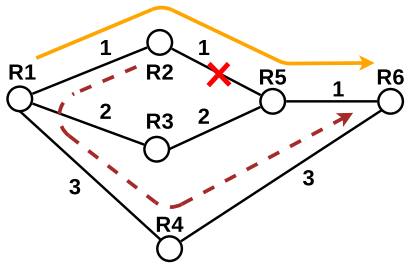

Fig. 20. Illustration of Not-via Addresses and the Failure Insensitive Routing with link costs as marked on the edges. Not-via: when link R2–R5 fails (marked by a red cross in the figure) along the shortest R1–R6 path (marked by an orange arrow), R2 will tunnel packets destined to R6 to the address "R6 not via R5". The packet will travel along the shortest R2–R6 path with the default next-hop R5 removed from the topology (marked by a dashed dark red arrow) immediately to the next-next-hop (i.e., R6). FIR: when link R2–R5 fails, R2 recomputes its routing table but will suppress the notification for the rest of the routers. Rather, it sends the packet back to R1, which, having received it "out-of-order" from R2, infers that link R2–R5 and/or R5–R6 has failed and hence forwards it via R4 to the destination.

shortest paths to rLFA candidates may converge along a single, possibly failed, next-hop. Not-via overcomes this problem by tunnelling/encapsulating packets through a special "not-via" IP address that explicitly identifies the network component that the repair must avoid. Hence, each router can pre-compute backup paths covering each individual failure case, by taking a modified graph from which the failed component was explicitly removed and then re-computing the shortest paths in this modified graph. In operation, any router along the detour, receiving a packet destined to a not-via address, immediately (1) knows that this packet is currently travelling on a detour and therefore the default routing table next-hop should not be applied, and (2) identifies the failed component associated with the not-via address and switches to the pre-computed backup shortest path. See Fig. 20 for an demonstration of the main concepts in Not-via.

Suppose that router s detects that its primary next-hop e towards router/prefix d has failed. In such cases, s encapsulates the packet in a new tunnel header and into the outer IP header it sets the destination address as a not-via address. The not-via address is an ordinary IP address that was administratively configured, and advertised into the IGP to mean "an address of router d that is reachable not via node e". In order to cut down the length of the bypass paths, in the original not-via specification the destination of the repair path is not immediately d but rather the "next-next-hop" of s to d (i.e., the next-hop of e to d). This behaviour has been questioned several times since then [200]–[202]. Routers in the IGP domain will advertise not-via addresses alongside the standard IP prefixes and store a next-hop in the forwarding table for each "x not-via y" address by (1) temporarily removing component y from the topology and (2) calculating the shortest path to x in the modified topology. As long as a failure does not partition the network, this mechanism ensures that each single component failure can be protected.

Notably, Not-via was the first viable IPFRR specification to guarantee full failure coverage against single-component failures. Yet, it remains fully compatible with the IP data plane (but not with the control plane, see below), requiring no expensive modification to IP router hardware. Unfortunately, the resource footprint may still be substantial. First, it requires maintaining additional not-via addresses, introducing significant management issues (no standard exists as to how to associate a not-via address with a concrete router/failure case), control-plane burden (possibly thousands of not-via addresses must be advertised across the IGP [203] and each potential failure case requires an additional run of Dijkstra's algorithm, and data-plane load (not-via addresses appear in the IP forwarding table, which already contains possibly hundreds of thousands of routes). In addition, tunnelling schemes, like rLFA and Not-via [130], [178], [204], [205] may raise unexpected and hard-to-debug packet loss or latency when the size of the encapsulated packets exceeds the MTU (Maximum Transfer Unit), causing a packet drop (when the "Don't fragment" bit is set) or a costly and timely fragmentation/reassembly process at the tunnel endpoints (in IPv4).

The Not-via specification sparked significant follow-up work. [206] introduces aggregation and prioritization techniques to reduce the computational costs of Not-via and the forwarding table size. The paper also proposes an algorithm (rNotVia) that allows a router to efficiently determine whether it is on the protection path of a not-via address and cuts down unnecessary calculations. Lightweight Not-via [200], [201] aims to break down the management burden, decrease the computational complexity, and reduce the number of not-via addresses needed, based on the concept of (maximally) redundant trees [202], [207]. This modification allows to cover multiple failure scenarios using a single not-via address, which reduces the number of necessary not-via addresses to 2 per router. The question whether the combined use of LFA and Not-via results in operational benefits is asked in [203]; the answer is generally negative.

Failure Insensitive Routing (FIR) using interface-specific forwarding is an IPFRR proposal originating from the academia, which gained significant following in the research community [173]. This is the first, and possibly the most elegant, IPFRR method proposed (the original paper appeared as early as 2003, predating even the first IPFRR draft, see Fig. 17), providing full protection against single link failures. Later versions and extensions address various shortcomings of FIR, e.g., provide protection against node failures ([176], [208]–[212], see below). Currently, we know of no off-the-shelf router products that implement FIR.

FIR is similar to Not-via in the sense that routers along a detour can identify the failed component. However, instead of using Not-via addresses that explicitly communicate the identity of the failure, in FIR routers rather *infer* it autonomously from packets' flight: when a packet arrives through an "unusual" interface, through which it would never arrive under normal operation, the set of potential links whose failure could lead to this event can be inferred. Using this inferred information, routers can divert packets to a new next-hop that avoids the failed links. This provides full failure case coverage, at the cost of the modification of the default IP data plane: forwarding decisions in FIR are made not only based on the destination address in a packet, but are also

specific to the ingress interface the packet was received on. See Fig. 20 for a sample topology demonstrating interface-specific routing.

The Failure Insensitive Routing (FIR) [173] scheme revolves around two basic concepts: *key links*, used to identify the potential failure cases that could lead to a router receiving a packet on a particular interface, and *interface-specific forwarding tables*, which assign a next-hop to each destination router/prefix separately for each ingress interface. Formally, given router s, destination d, and next-hop e from s to d, the key links at s with respect to d for the interface e-s are exactly the links along the $s \rightarrow d$ shortest path except s-e. The interface specific forwarding table with respect to this interface is obtained simply by removing the key links from the topology and running Dijkstra's algorithm from s. As long as the topology is 2-connected, removing the key links will not partition the network and full failure coverage is attained.

Once computed, a router will install the per-interface forwarding tables into the data plane and, in normal operation, use standard shortest-path routing to forward packets. To deal with remote failures, the router does not need to react as the interface-specific routes are specific enough to handle all detours, while for local failures it recomputes its shortest paths but suppresses the IGP failure notification procedure from this point in line with the requirements of the IPFRR framework.

In a nutshell, in FIR a router detects that a packet is on a detour from that it is received from its primary next-hop. In this regard, FIR resembles U-turn alternates but it is much more generic than that (works for alternates more than two-hops away). As we have seen, it also generalizes Not-via in that it does not need additional not-via addresses to identify the failure, it can infer this information.

FIR is remarkably elegant, simple, and fully distributed, and it also provides full failure case coverage, including node failures (see below). On the negative side, interface-specific forwarding is still not available in the standard IP data plane: while most 3rd generation routers store a separate forwarding table at each line card, management and monitoring APIs to these per-interface forwarding tables have never been standardized. In addition, FIR may create persistent loops when more than one link simultaneously fail. To address these issues, [213] extends FIR to handle node failures, [176], [209] generalizes this method to asymmetric link costs and inter-AS links, [210] provides a version that is guaranteed to be loop-free at the cost of increasing forwarding path lengths somewhat, while [208], [211], [214] present further modifications to the backup next-hop selection procedure based on interface-specific forwarding to handle, among others, double link failures. For more recent improvements to IPFRR, see [195].

C. Overlay-Based Reroute

Achieving full failure case coverage in IPFRR and adhering to IP's default connectionless destination-based hop-by-hop routing paradigm at the same time seems a complex problem. Each shortest-path-based IPFRR method we described above suffers from one or more significant shortcomings due to this fundamental contradiction, in terms of failure case coverage or deployability (or both).

The IPFRR methods based on the *fault isolation* technique described below offer a way out of this problem: they implement certain overlay networks on top of the physical topology so that for every possible local failure there will be at least one overlay unaffected by the failure that can be used as a backup. Here, an IP overlay may be any virtual IP network layer provisioned on top of the physical IP infrastructure using, e.g., multi-topology routing or virtual routing or forwarding (VRF) instances. Of course, this may break IP compatibility, as some way to force bypass traffic into the proper overlay is needed: most proposals use some additional bits in the IP header (e.g., in the ToS field), but tunnels, not-via addresses, and multi-topology routing could also be reused for this purpose. This allows bypass paths to no longer be strictly shortest, which lend more degrees of freedom to assign detours for different failure cases and leads to higher failure case coverage.

Below, we summarize the most important fault-isolation techniques proposed so far, with the emphasis on the methodology to obtain the overlay topologies themselves. In the taxonomy of Section II-B, the fault-isolation-based IPFRR mechanisms discussed below (not to confuse with fault detection and fault localization, see Section II-C) belong to the class of *preplanned, shared or dedicated, global protection* schemes.

General Resilient Overlays is a class of IPFRR methods that use different, non-shortest-path-based algorithmic techniques to compute the backup overlays. These methods will be called "general" to stress that the backup can be an arbitrary topology, differentiating this class from the subsequent class where the backups will be strictly limited to tree topologies.

The authors in [215] trace back general fault isolation to FRoots, which considered the problem in the context of high-performance routing [216]. The idea in that paper, and in essentially all extensions [174], [175], [177], [215], [217]–[221], is the same: find a set of overlay topologies so that each router will have at least one overlay it can use as a backup whenever its primary next-hop fails. The difficulty in this problem is to consider arbitrary, possibly highly irregular network topologies and/or link costs [216], and to minimize the number of the resultant overlays to cut down control, management, and data-plane burden.

Key to all IPFRR methods described in this class is the notion of *failure isolation* and *backup overlays*. We say that a topology (e.g., an overlay) isolates a failure when the topology, and the routes provisioned on top of the topology, have the property that there is a certain set of source-destination pairs (the "covered pairs") and failure cases (the "isolated failures") so that if we inject a packet into the topology at a covered source towards a covered destination, the packet will avoid the isolated failure and flow undisrupted to the required destination. The set of backup overlays then forms a family of overlay topologies, with the property that for any pair of a source and a destination router and any failure there is at least one backup overlay that isolates the failure and covers the source-destination pair. Then, if the source selects the proper backup overlay for the given failure case(s) and destination

node and it has a way to "pin" the packet into that overlay, then it is ensured that the packet will reach the destination node avoiding the failure(s).

Provided that a way exists to pin packets into an overlay, this family of IPFRR mechanisms does not require major modification to the IP control plane and data plane and hence is readily deployable, although not in an incremental way. Also, fault isolation may yield 100% failure case coverage, even for multi-failures [218]. On the other hand, depending on the network topology and link costs the number of backup topologies may be large (in the order of dozens), raising a substantial management issue. This is especially troublesome when the task is to protect against multiple failures [219]. Additionally, pinning packets to the overlay may not be simple, as the IP header space is a valuable yet scarce resource and using some bits for tagging the backup overlay may interfere with other uses of these bits (e.g., tagging in the ToS field clashes with the use of the same field in DiffServ). Also, some non-standard technique is needed by routers to inject packets into the correct overlay (but see VRO below). Finally, finding the fewest possible backup overlays is usually a computationally intractable problem, which calls for approximations that may deteriorate efficiency.

Starting from FRoots [216], there are many realizations of the basic general fault-isolation technique. Below, we survey the most representative ones. The earliest proposal (following FRoots) seems to be Resilient Routing Layers [174], where the backup overlays are obtained by explicitly removing certain links from the overlay topologies. If the overlays are such that each link is isolated by at least one overlay, then full protection against single-link failures can be attained. Multiple Routing Configurations (MRC) and the related line of works [175], [215], [217], [218] took another approach: instead of effectively removing links from the overlays they reach isolation by setting link costs separately in each overlay so that shortest paths will avoid a certain set of links, effectively isolating these failure cases. This method is easy to adopt for link-, node-, and multiple-failure protection scenarios [218]. Perhaps the most viable implementation avenue was outlined in [222] (but see also [219], [223]): the use of multi-topology routing. Multi-topology routing is a feature that is to become available in IGPs [224] to maintain multiple independent overlay routing topologies on top of the default shortest paths, with each topology using an independent set of link costs. One routing topology is devoted to support IP forwarding under normal networking conditions, and when a next-hop of some router becomes unavailable it simply deviates all traffic that would have been sent via that next-hope to another interface, over an alternative routing topology. With a clever optimization of the link costs per topology, full failure case protection against link and node failures can be achieved [219], [222], [223]. The Independent Directed Acyclic Graphs proposal [220] uses a set of DAGs for similar purposes.

Most recently, *Virtual Routing Overlays* (VRO [177], [221]) solves two problems inherent in failure isolation, namely, the lack of standards for injecting packets into overlays and for pinning a packet into the an overlay until it reaches the destination. VRO uses LFA and virtual routers for this purpose: it provisions multiple virtual routers on top of each physical router, each instance with its own set of local link costs, so that whenever a packet encounters a failure it will automatically fall back to a virtual router that is provisioned as an LFA and will flow within the virtual overlay until it reaches the destination. The authors in [221] are able to show that per each physical router at most 4 virtual router instances in the worst case, and at most 2 instances in the average case, suffice to reach 100% failure case coverage.

Independent/redundant trees are a class of IPFRR methods based on the fault isolation technique that restrict the backup overlays to be trees [154]–[156], [207], [225]–[228].

The idea of searching for backups in the form of simple tree topologies traces back to redundant trees, red-blue trees, and ST-numberings [202], [207]. Consider the example of *redundant trees*, for instance: here, [207] shows a polynomial time algorithm to find a pair of mutually link-disjoint directed trees towards any node (the "red" and the "blue" tree) and claims that such pair of paths can be found in each 2-connected graph. Adopting this idea to IPFRR is then straightforward: whenever the primary next-hop is on the red tree and this next-hop fails then we pin packets to the blue tree (and *vice versa*): the art is then to implement this scheme given the limitations of the IP data plane [201]. *Independent trees*, in contrast, are generally undirected, serving as a backup to potentially multiple destination nodes [227], but in general they may be more difficult to compute [229].

Tree topologies have the property that there is at most one path between any pair of nodes in the tree. Searching the backup overlay in the form of a tree therefore solves several issues in one turn: it prunes the search space significantly, yielding that in most cases the backup trees can be found in polynomial time (recall, the general question is usually NP-complete) and makes routing on top of the overlay trivial. The price is, of course, an increased number of backup overlays needed to be provisioned: while the number of general overlays grows very slowly with the number of routers (close to logarithmically, at least by experience [215]), redundant trees are per-node and hence the number of overlays scales linearly with the number or routers in general [154].

The pros are that independent trees are easy to calculate, can be generalized for multiple failures, and forwarding along the tree-shaped overlay is trivial. On the negative side, we may still need to pin packets to the correct overlay some way, which may cause compatibility issues. Furthermore, circular dependencies between backup trees may make it difficult to find the correct backup in case of multiple failures [202].

Thanks to the simplicity of this approach, there are many versions, extensions, and even standards that use independent/redundant trees for IPFRR [154], [155], [179], [200]–[202], [225]–[228]. The underlying mathematical notions, revolving around red-blue trees, ST-numberings, and independent trees, and the conditions under which they exist, are generally well-understood [179], [202], [229]. Proposals then differ based on whether they use directed (redundant) trees or independent (undirected) trees, and the different ways they implement the resultant overlays.

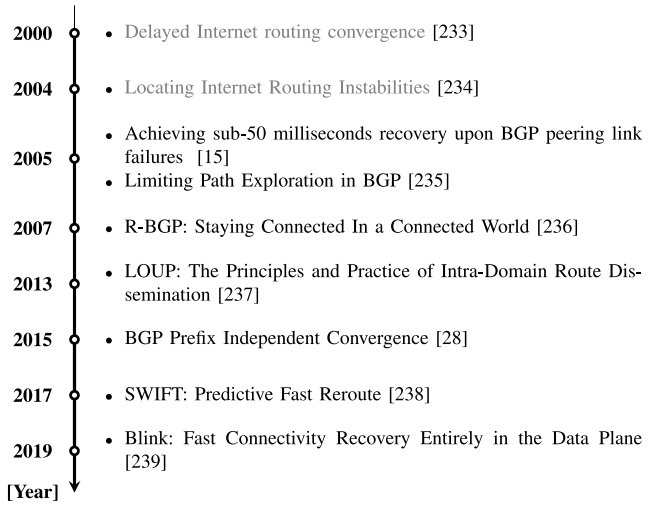

Fig. 21. Timeline of the selected documents and solutions related to inter-domain network-layer fast recovery (entries marked in gray provide the general context).

The earliest proposal to use redundant trees for IPFRR is Lightweight Not-via [200], [201], whereby packets are pinned into the proper (red or blue) overlay tree using not-via addresses. After multiple modifications, this proposal, and the related concept of maximally redundant trees for non-2-connected topologies, reached RFC status and became the MRT-FRR standard [154], [155], [225]. Recently, [179] proposes various heuristics to cut down the length of the resultant paths and [230] compares the performance of MRT-FRR to that of not-via addresses. Similar ideas exist to reach the same goals using undirected independent trees for handling multi-failures [212], [226]–[228].

D. Summary

In this section, we reviewed the most important ideas, proposals, and standards for providing fast network-layer failure recovery in the intra-domain setting. We reviewed IPFRR, the umbrella framework for fast local IP-layer protection, in Section V-A. Then, we discussed the prevalent schemes in this context, namely, the methods that reuse the IGP's link state database maintained and shortest-path routing for failure recovery (Section V-B) and the schemes that revolve around virtual overlay networks on top of the default forwarding topology for this purpose (Section V-C).

In general, fast IP failure recovery in the intra-domain unicast setting is a well-understood problem area, with several well-established standards [32], [130], [150], [154], [155], [169], [178], [190], [225], inter-operable implementations in off-the-shelf routers [181]–[183], [231], operational experience [180], [203], and extensive research literature [173], [184] available. Open problems remain, however: even after several iterations it is still not entirely clear how to reliably avoid intermittent micro-loops without slowing down re-convergence [190] and provide the required quality of service even during routing transients [152], [153].

Finally, we note that the vast address space that becomes available with the deployment of IPv6 opens up completely new possibilities for IP fast recovery. A prominent example is the emerging segment-routing framework [197], a variant of source routing where an explicit route can be encoded into packets as a sequence of smaller path chunk called *segments*. Apart from the new perspectives on network programming and traffic engineering, the standardization of the IPv6 Segment Routing Header [232] enables new fast-recovery schemes to attain complete failure recovery on top of a pure IPv6 data plane; see, e.g., [199].

VI. Inter-Domain Network-Layer Fast Recovery

Realizing FRR in the inter-domain routing, i.e., across network domains administered by independent entities, entails solving an additional set of challenges compared to intra-domain FRR. The Border Gateway Protocol (BGP) is today's de-facto inter-domain routing protocol that dictates how network organizations, henceforth referred to as *Autonomous Systems* (ASes), must exchange routing information in the Internet to establish connectivity. Intuitively, since networks do not have control and/or visibility into other network domains, the proposed FRR techniques often achieve much lower resiliency to failures than intra-domain mechanisms. Moreover, inter-domain FRR often requires some sort of coordination among network domains in order to guarantee connectivity, which has been an obstacle for most of the proposed schemes, preventing them from being standardized. In this section, we discuss challenges around the problem of realizing FRR in the wider Internet (i.e., in BGP) and we describe different approaches proposed to tackle this problem. We present a timeline with a selection of work on the topic in Fig. 21, starting from the initial work in the early 2000s exposing the problem of slow BGP convergence, and then covering the sequence of work on improvements for the detection of failures and fast restoration of connectivity in the Internet.

A. Background on BGP Route Computation

BGP is a per-destination policy-based path-vector routing protocol. The owner of an IP prefix π configures its BGP routers to announce a route destined to π to (a subset of) its neighbors according to its routing policies. A BGP route contains the destination IP prefix destination, the sequence of traversed ASes (i.e., the `AS_PATH`) and some additional information that is not relevant for this section. When an AS receives a new BGP route, it extracts the destination IP prefix destination π and checks whether the currently selected route is worse than the new one based on its routing ranking policies. In this case, it updates its new best route towards π and announces it to (a subset of) its neighbors according to its routing policies. In the case of link/node failure, reconvergence is triggered by the node adjacent to the failure, which either selects an alternative best route towards the destination or withdraws the existing route by informing its neighbors.

B. Challenges

Realizing Internet-wide FRR requires taking into consideration three unique aspects of the inter-domain routing setting. First, the Internet is not administered by a single organization, but it rather consists of thousands of independent

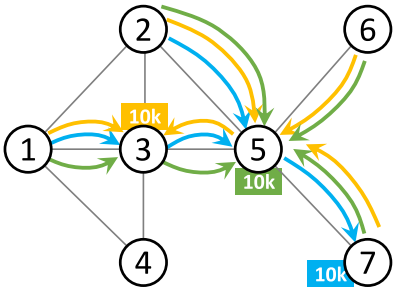

Fig. 22. An illustration of SWIFT: the pre-failure state.

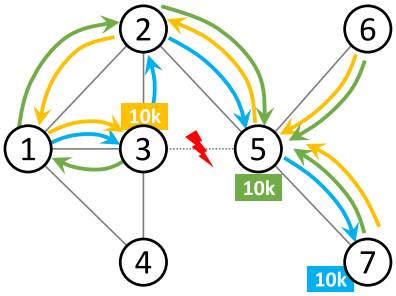

Fig. 23. An illustration of SWIFT: the post-failure state.

interconnected network organizations with possibly conflicting goals. Second, BGP convergence is significantly slower than intra-domain routing protocols (i.e., on the order of minutes compared to hundreds of milliseconds/seconds for intra-domain protocols) [233], [238]–[242]. Finally, Internet routing protocols must guarantee connectivity towards hundreds of thousands of destination IP prefixes. There are four main consequences related to the design of FRR mechanisms that we can draw from the above considerations. First, failures of inter-domain links can lead to long and severe disruptions, as the BGP control-plane may take minutes to reconverge. FRR at the inter-domain level is therefore critical for restoring Internet traffic. As the number of prefixes can be on the order of tens of thousands, being able to update the forwarding tables quickly is of paramount importance. For example, the reported results of previous measurements indicate that it can take several seconds to update tens of thousands of prefixes [238]. Second, one cannot assume that all ASes will deploy FRR mechanisms. This makes clean-slate FRR approaches very difficult to realize in practice. Third, a network operator has no visibility of the entire Internet network. Some ASes may end up (mistakenly or on purpose) detouring their traffic away from their announced BGP routes without communicating this information to their BGP neighbors [243]. Fourth, BGP does not carry any root-cause analysis information within its messages (i.e., no explicit link failure notification is available), since all networks must jointly agree on the format of BGP messages, making it hard to be modified. Fifth, inter-domain routing messages are disseminated throughout a domain on top of intra-domain routing protocols. Such an overlay makes the prompt restoration of inter-domain connectivity even more complex [237].

Given the above challenges, we first describe two critical problems concerning the support for FRR on the Internet that go beyond the computation of backup routes: detection of possible remote failures and quick updates of the forwarding planes. We then discuss a set of improvements to BGP that would allow network operators to precompute backup Internet BGP routes.

C. Detection of Local and Remote Failures

Detecting failures in the inter-domain routing entails solving different challenges than in intra-domain routing. First, to implement FRR in BGP, beyond detecting local failures, an AS needs to detect and localize remote Internet failures, as *i)* downstream ASes may not perform any FRR and *ii)* BGP is slow at recomputing a new route, thus leading to traffic disruptions lasting even several minutes. Failures of adjacent peering links between two ASes can be detected using traditional techniques (e.g., BFD [78]) and are not discussed here. Detection of remote link failures can be performed both at the control plane and at the data plane, and we discuss these two different approaches in the following parts of this section.

Detection at the Control-Plane Level: At the control-plane level, a variety of techniques to detect and localize network failures using BGP data have been proposed. Some techniques require coordination from a geographically-distributed set of trusted vantage points, e.g., PoiRoot [234], [244], [245]. They are typically general enough to detect a variety of anomalies, including link failures and routing instabilities, and root-cause analysis. On the other side of the design space, we have techniques that do not require any coordination among ASes and simply attempt to infer remote link failures from the received bursts of BGP messages describing the route changes/withdrawals at one single location. We discuss more in details these types of detection mechanisms as the detection can be performed locally at the specific node/network that reroutes traffic.

We focus on the SWIFT system [238] and how it detects remote failures on the Internet. SWIFT performs the root-cause analysis on a sudden burst of BGP updates using a simple intuition: it checks whether a certain peering link has suddenly disappeared from the `AS_PATH` of a certain number of BGP announcement messages received close to each other. Internally, SWIFT relies on a metric called *Fit Score* and infers the failed link as the one maximizing its value. Consider the example shown in Fig. 22 and Fig. 23 inspired by the original paper, where we have seven AS nodes numbered from 1 to 7. `AS 3`, `AS 5`, and `AS 7` originate ten thousand distinct IP prefixes each, marked in orange, green, and blue, respectively. The converged state of BGP is shown in Fig. 22 (orange, green, and blue arrows represent the BGP paths associated with the corresponding IP prefixes). We assume that `AS 2` does not announce any green route towards `AS 5` to its neighbor `AS 3`. Let us now consider `AS 1` which is sending traffic towards `AS 7` along path (1 3 5 7). When link (3, 5) fails (see the post-convergence state in Fig. 23), `AS 1` starts receiving a sequence of at least twenty thousand updates for all the routes towards `AS 5` and `AS 7` that traverse link (3, 5). `AS 1` therefore infers that one of the peering links from among (1, 3), (3, 5), and (5, 7) must have failed. To determine which link is down,

AS 1 waits a little bit to see which paths are initially being modified by BGP. Since AS 1 does not see any path change for the IP prefixes destined to AS 3, it quickly infers that link (1, 3) remains fully operational. Moreover, since AS 1 will soon receive a new route from AS 3 towards AS 7 via link (5, 7), it infers that the failed link must be (3, 5).[6] After identifying this link, SWIFT quickly reconfigures all the routes traversing link (3, 5), so that they follow a backup route (see the following subsection). We note that the inference algorithm used in SWIFT is more sophisticated than the simplified version that we have just presented here. In particular, for each link l at time t, SWIFT computes the value of the *Fit Score* metric which tracks *i)* the number of prefixes whose paths traverse link l and have been withdrawn at time t, divided by the total number of withdrawal messages received until t, and *ii)* the number of prefixes whose paths traverse link l and have been withdrawn at time t, divided by the total number of prefixes whose paths included link l at time t. Based on the *Fit Score* metric, SWIFT is able to infer the status of a remote link.

One definite advantage of fast-recovery mechanisms inferring the status of the network from control-plane messages is that they significantly speed up the recovery upon a failure. At the same time, one of the main disadvantages is that tuning the parameters of these mechanisms for each and every network is often difficult. In particular, general and comprehensively-validated guidelines are still missing. Moreover, deployment of SWIFT-like mechanisms in independent network domains may lead to severe transient forwarding loops due to the uncoordinated nature of SWIFT in situations when multiple failures arise.[7] Finally, control-plane approaches are inherently limited by the slow reconvergence of BGP.

Detection at the Data-Plane Level: At the data-plane level, detection of remote failures can be performed by *i)* monitoring explicit application performance-related information (e.g., similarly to Google Espresso [246]), *ii)* sending active probes (similarly to BFD) [245] to verify the route followed by a packet and to check whether the route is valid, or *iii)* passively inspecting the transported traffic to detect if a subset of the flows are retransmitting a non-negligible number of packets, as described in the Blink system [239]. In particular, Blink uses properties of the TCP implementation/TCP stack which retransmits a non-ACK'ed packet exactly after 200 ms.[8]

One advantage of data-plane approaches is the speed of the failure detection process, as traffic travels order of magnitudes faster than control-plane messages. However, one clear disadvantage is that identifying the location of a failure is a much harder problem, which requires active probing (e.g., traceroutes) and cannot be inferred solely from data traffic.

D. Updating the Forwarding Tables

Single link failures between two ASes may affect hundreds of thousand of destination IP prefixes. Updating the forwarding plane to reroute all these prefixes is therefore a critical operation to achieve fast restoration. For example, assuming 100 μs to update a forwarding entry [247], it takes roughly 10 seconds to update a forwarding table containing 100 thousand prefixes.

Consider the example shown in Fig. 24a in which we depict the forwarding table of a BGP router located in AS 1. The router has installed forwarding rules for the ten thousand IP prefixes originated by AS 3 (the first 10K prefixes), AS 5 (the second 10K prefixes), and AS 7 (the third 10K prefixes). For the sake of simplicity, we denote BGP next hops using AS identifiers, e.g., the BGP next hops of AS 1 are AS 2, AS 3, or AS 4.

When a packet must be forwarded from AS 1, a lookup is performed to extract the next hop towards which the packet is forwarded. This data-plane design clearly has the benefit of achieving low packet processing time (one lookup) and low memory occupancy (one entry per prefix) and has long been used as the reference data-plane architecture by the main router vendors such as Cisco [248], [249] in the early 2000s. Yet, this approach is problematic upon peering link failures/flapping. When the link (1, 3) fails, the router must rewrite the first, third, and fourth 10K thousand forwarding entries in its table to steer traffic away from AS 3. We assume that traffic destined to AS 3 will be rerouted through AS 4 while the remaining traffic destined to AS 5 and AS 7 will be rerouted through AS 2. Rewriting 30K entries is an operation that takes non-negligible time and during which traffic will be dropped. The update time in this case grows linearly in the number of prefixes.

This problem is less severe in intra-domain routing where the number of destinations is on the order of few hundred.[9] Solutions to speed-up the update of the forwarding tables upon a failure have been proposed and we divide them into those targeting failures of adjacent (local) AS-to-AS links and remote ones.

BGP-PIC [28], [247]: We now discuss a technique to speed up data-plane updates due to an adjacent inter-domain link failure. In the case of a local peering link failure, one can associate in the data plane a Virtual Next Hop (VNH) with each destination IP prefix, and the VNH with the actual BGP next hop. This VNH simply acts as an *indirection* between the IP destination and the real next hop. Two IP prefixes are associated with the same VNH if and only if they share the same primary and backup AS next hops. Intuitively, one can update many prefixes associated with the same VNH by simply updating the assignment of the VNH to the real AS next hop. This technique has been presented in 2005 [247] and has been incorporated in Cisco devices with the name BGP PIC [28] years later.

We describe the BGP-PIC mechanism through an example. Consider the example shown in Fig. 24b. For each prefix, the control plane at AS 1 computes both the best and backup BGP best routes (and BGP next hops). It then groups IP prefixes that

[6]Note that the link failure inference algorithm implemented in SWIFT runs on a per-BGP-session basis and it does not combine BGP updates received from different sessions.

[7]Note that SWIFT has some guaranteed properties — see the original paper for a detailed discussion [238].

[8]Common value used in most implementations on Linux systems.

[9]Hierarchical intra-domain routing is a common solution to scale both the routing computation and the time needed to update the forwarding state. In contrast, the only hierarchical optimization in BGP is the aggregation of the IP addresses to compute BGP routes and the usage of Longest-Prefix-Matching techniques to forward packets at the data-plane level.

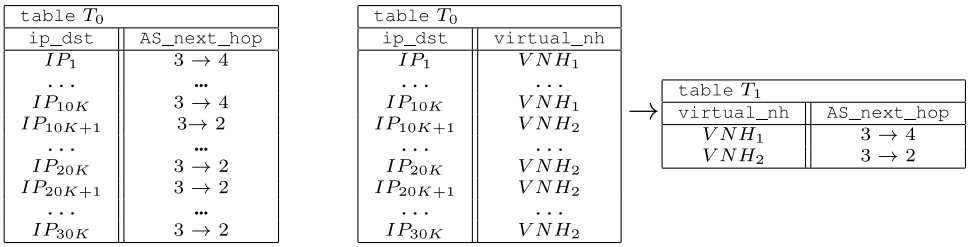

Fig. 24. Fast data-plane convergence: (a) classical data-plane architecture and (b) the equivalent simplified BGP-PIC data-plane architecture. We indicate updates of single forwarding entries with arrows depicted inside the field of the forwarding entry.

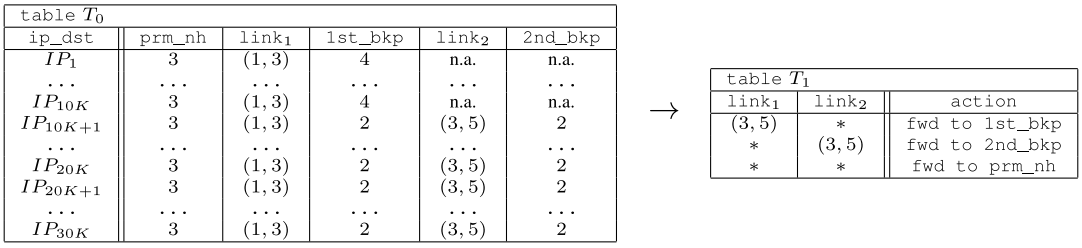

Fig. 25. Simplified SWIFT data-plane forwarding table at AS 1.

have the same forwarding primary and backup routes in Virtual NextHops (VNHs). In our example, we have just two VNHs: VNH_1 which contains prefixes $P_1, \ldots, P_{10K}$ whose primary route is through AS 3 and backup route through AS 4, VNH_2 which contains prefixes $P_{10K+1}, \ldots, P_{30K}$ whose primary route is through AS 3 and backup route through AS 2. When a packet must be forwarded at AS 1, the first lookup determines the VNH of the packet, while the second lookup determines the actual BGP next hop.

The main benefit of this approach is that the time needed to update the forwarding table upon the failure of link (1, 3) is greatly reduced. In fact, as soon as AS 1 learns that link (1, 3) is down, it simply updates the next-hop of VNH_1 to AS 4 and it updates the next-hop of VNH_2 to AS 2: two single updates of the forwarding table. In the worse case, in a large network with *N* BGP border routers, this approach may require $n - 1$ data-plane updates, which may be performed in roughly 1 ms when $n = 100$. One remaining problem with BGP-PIC is that it only works for adjacent link failures and cannot be easily generalized to remote link failures.

SWIFT [238]: In the case of a remote link failure, one can generalize the above approach, i.e., create a different VNH for each different AS_PATH. Upon identification of a failure, the control plane can update the VNH mapping based on the computed backup information and only update the VNH traversing the failed link. The number of forwarding table modifications is linear in the number of VNH traversing the failed links, possibly a large number. A different approach has been presented in SWIFT, whose remote failure detection mechanism has already been discussed in Section VI-C. We present a simplified description below, focusing only on the essential insights related to SWIFT.

The control plane associates a backup AS next hop with each destination IP prefix and an inter-AS link traversed on the way towards the destination. The alternative next hop can be quickly activated upon detection of a remote link failure using a carefully designed packet processing pipeline like the one described in the SWIFT system [238] and summarized below. We describe the SWIFT data-plane with an example. Fig. 25 shows the forwarding table based on the example from Fig. 22. The SWIFT forwarding pipeline consists of two tables, T_0 and T_1, arranged in a sequence. Table T_0 maps each IP address to a sequence of actions that fetch information about the traversed ASes as well as the primary and backup links for the associated IP address. In the example, we only protect against failures of the (local) first AS link and the (remote) second AS link. One can protect against a larger number of links by adding more remote links. For each link, we store the backup AS next hop. The first table attaches all the backup options to the packet header. For instance, a packet destined to IP_{10K+1} will have AS 2 as a backup if link (1, 3) or link (3, 5) fails. Table T_1 is used to determine the correct next-hop AS by verifying whether an entry exists in T_1 that matches any of the backup information attached to the packet. More specifically, when a link fails, the control plane installs two entries in Table T_1, both matching the failed link and instructing the device to forward the packet to the corresponding backup AS. For instance, if link (3, 5) fails, the control plane will install two rules that match all packets traversing link (3, 5) at either the first or the second hop, so that the matching packets can be forwarded towards the corresponding backup AS next hop. The implementation of the action in Table T_1 depends on the expressiveness of the data plane. For instance, in OpenFlow, one needs to add rules matching each possible backup next hop to the forwarding action towards that backup next hop. In P4, however, this action could be expressed in a simpler way by extracting the backup next hop from the metadata and finding the egress port based on this next hop.

One advantage of SWIFT is that it requires few updates to the forwarding table to detour the forwarded traffic to a different next hop. More specifically, the number of updates to the forwarding table is linear in the length of the longest path

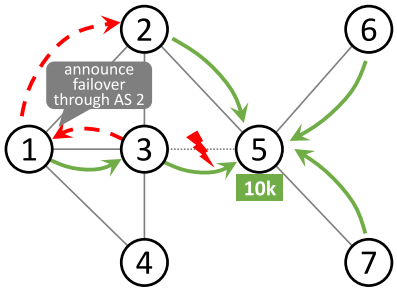

Fig. 26. R-BGP: AS 1 advertises a backup route to AS 3, offering to transit its backup traffic to AS 5 through AS 2.

that a network operator wants to protect, e.g., two updates to protect the first and second AS links for all the IP prefixes. At the same time, updating the backup information may still require a large number of updates (these updates do not affect the forwarded traffic, though), thus increasing the overhead on the operating system of the switch. Another limitation of SWIFT is that it only provides a fast-reroute alternative for single-link failures assuming there exists an already available BGP route that is not traversing the failed link. Computing BGP routes that are robust to any link failure is beyond the scope of SWIFT and will be discussed in the next section.

E. Fast-Reroute Mechanisms

BGP is a per-destination routing protocol and, as such, comes with limitations similar to IP FRR techniques discussed in Section V. Specifically, for some network topologies, it is impossible to find per-destination FRR mechanisms that are robust to even single link failures. Consider the forwarding state shown in Fig. 26 where the green solid arrows represent the forwarding before link (3, 5) fails while the red dashed arrows represent the forwarding at AS 1 and AS 3 after the link has failed. Note that we assume AS 2 does not announce any route towards AS 3. This means that, if the link between AS 3 and AS 5 fails, AS 3 is left with no safe neighbor to whom it could forward its traffic (a forwarding loop would be created otherwise by forwarding traffic back to AS 1).

Inter-domain FRR mechanisms, such as Blink [239], SWIFT [238], and Google Espresso [246], simply reroute traffic along any of the available BGP routes. While these mechanisms cannot guarantee robustness to even single link failures, these approaches are legacy-compatible with BGP and, at the same time, provide some minimal degree of robustness. We now discuss the BGP-based FRR mechanism called R-BGP [236] for handling single link failures over the Internet without the need to wait for BGP reconvergence.

R-BGP is an enhancement of BGP that allows ASes to compute backup paths between ASes in a distributed manner. In R-BGP, ASes announce one additional backup route towards their downstream neighbors, which, in turn, can use this backup route in the event of a downstream failure. One key idea of R-BGP is to limit the amount of extra messages that are needed to be exchanged over BGP by leveraging the unique properties of how BGP routes propagates through the Internet according to customer-peer-provider relationships.

We describe R-BGP through an example. Consider the example in Fig. 26. AS 1 now advertises a backup route to AS 3, offering to transit its backup traffic through AS 2 for all the prefixes announced from AS 5. When link $(3, 5)$ fails, AS 3 simply sends all its traffic to AS 1, which in turn detects from the fact that the traffic is received from an "outgoing" direction that the traffic should be forwarded along the backup path. It therefore immediately sends traffic to AS 2.

The above mechanism, described in detail in the original paper, guarantees connectivity for any single link/node failure at the inter-domain level. The authors of the paper describe a sequence of optimizations to reduce the amount of additional information piggy-backed on BGP and how to handle spurious withdraw messages (see Section X for more information). The authors also note that special care during the BGP reconvergence process must be taken into account by *incorporating* information about the root-cause of a failure into BGP. The main disadvantage of R-BGP is therefore that it requires modifications to the BGP protocol that must be adopted by a large number of networks, a cumbersome operation in practice. Finding ways to enhance R-BGP to handle multiple link failures and make it incrementally deployable are natural future research questions that have remained open.

F. Summary

In this section, we reviewed the most prominent techniques to deal with failures at the Internet level. We first discussed the main differences in dealing with inter-domain fast reroute compared to intra-domain: (1) the lack of control and visibility into the entire Internet topology and (2) the large-scale amount of destination IP prefixes to be handled. We therefore discussed the main approaches to *quickly* detect failures at the inter-domain levels by inferring such failures from both BGP-based control-plane signals as well as TCP-based data-plane ones [238], [239]. We then discussed techniques to quickly update the forwarding plane when thousands of forwarding rules have to be modified in response to a link failure. Such techniques either rely on (1) what we referred as"indirection" tables for mapping virtual identifiers to a specific forwarding actions [28], [247] or (2) labeling destination IP prefixes with their explicit path and matching that path with the failed link to compute the backup next-hop [238]. We concluded the section by discussing the currently available yet simple FRR mechanisms in BGP, i.e., reroute on any alternative existing path, as well as mechanisms that require modifications to BGP but achieves guaranteed resiliency for every single link failure [236].

Many open research problems still require to be addressed: despite decades of academic and industrial efforts in improving the resiliency of BGP, the current status-quo is still quite alarming. Substantial efforts must be targeted to the problem of detecting failures at the inter-domain level as both Swift [238] and Blink [239] suffer from false positives and false negatives. Moreover, data-plane approaches such as Blink are highly dependent on the specific configuration of RTO timeouts, which makes it both inaccurate when different congestion control mechanisms will be deployed, but also vulnerable to malicious attacks. Finally, supporting backup paths in BGP

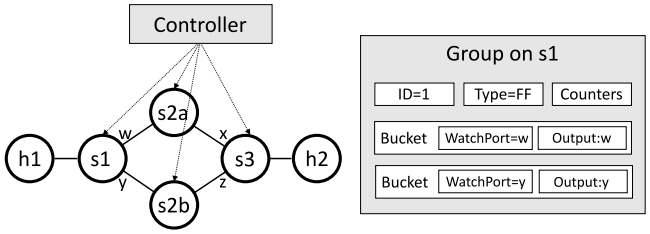

Fig. 27. Illustration of OpenFlow group table: fast-failover (ff) group table for `s1`.

seems to be a non-trivial challenge because of the inherent needs to preserve legacy-compatibility, privacy, and the sheer size of routing information currently exchanged on the Internet to glue almost hundreds of thousands of networks and almost one million IP prefixes.

VII. Fast Recovery in Programmable Networks

In this section, we discuss advanced fast-rerouting mechanisms as they are enabled by emerging programmable networks. We first provide some motivations for leveraging programmable network approaches with the goal of improving network robustness. We then discuss the two main generations of programmable specifications, i.e., OpenFlow and the most recent P4. Beyond discussing the existing body of work, we show through a practical example how the additional programmability of the P4 language provides unparallel opportunities for improving fast reroute mechanisms.

A. Motivation and Background

A simpler and faster failure handling was one of the reasons behind Google's move to Software-Defined Networks (SDNs) [250]. In an SDN, the control over the network devices (e.g., OpenFlow switches) is outsourced and consolidated to a logically-centralized controller. This decoupling of the control plane from the data plane allows to evolve and innovate the former one independently of the constraints and lifecycles of the latter one.

However, it also introduces new challenges. If a link failure occurs, it needs not only be detected but also communicated to a controller which then reconfigures affected paths. This indirection does not only introduce delays, but if based on in-band signaling, the network elements and the controller may even be disconnected due to the failure. For example, controller reaction times on the order of 100 ms have been measured in [251]: the restoration time also depends on the number of flows to be restored, path lengths, traffic bursts in the control network, and may take even longer for larger networks.

Failover in the data plane is hence an even more attractive alternative in the context of SDNs. Local fast-reroute allows an SDN switch (or "point of local repair") to locally detect a failure and deviate affected traffic so that it eventually reaches its destination. In the following, we first discuss solutions based on OpenFlow [18], the de-facto standard of SDN. Subsequently, we discuss solutions for programmable dataplanes, such as P4.

B. Recovery Concepts in OpenFlow

OpenFlow supports basic primitives to implement fast failover functions. In particular, OpenFlow 1.1.0 provided a fast-failover action which was not available before: it incorporates a fast failover mechanism based on so-called *groups* allowing to define more complex operations on packets that cannot be defined within a flow alone [252]. And in particular, to predefine resilient and in-band failover routes which *activate* upon a topological change. Before the introduction of such FRR primitives, researchers relied on ad-hoc non-standardized extensions to OpenFlow, e.g., flow priorities, timers, automatic deletion of rules forwarding on failed interfaces, to implement FRR primitives [253].

Fig. 27 illustrates the OpenFlow model and its FRR mechanism: a controller (e.g., Floodlight) can populate the forwarding tables of the different switches from a logically centralized perspective. To support fast-failover to alternative links without control plane interaction, OpenFlow uses group tables: the controller can pre-define groups of the type fast-failover (FF in the figure, the group table of $s1$ is shown). a group contains separate lists of actions, referred to as *buckets*. The fast-failover group is used to detect and overcome port failures: each bucket has a watch port and/or watch group to watch the status of the indicated port/group. The bucket will not be used if the element is down. That is, a bucket in use will not be changed unless the liveness of the currently used bucket's watch port/group is updated. In this case, the group will quickly select the next bucket in the bucket list with a watch port/group that is up. The failover time here hence depends on the time to find a watch port/group that is up.

In principle, most of the FRR mechanisms discussed earlier in this paper, e.g., for MPLS or IP networks, could be ported to SDN. However, without additional considerations, this approach could lead to an unnecessarily large number of flow table entries or to suboptimal protection only [262]. The first is problematic, as the flow table size of OpenFlow switches are often small so that only a few additional forwarding entries can be accommodated for protection purposes. The second is not acceptable for SDN, because unprotected flows remain disconnected or FRR-caused loops persist until the controller comes to rescue.

Several solutions based on OpenFlow have been proposed in the literature so far, e.g., based on ideas from LFA [192], [264] or MPLS [254], by encoding primary and backup paths in the packet header [255], or by extending to OpenFlow's fast failover action based on additional state in the OpenFlow pipeline: for example, *SPIDER* [260] leverages packet labels to carry reroute and connectivity information. An alternative OpenFlow switch design which relies on logical group ports and which aims to support connectivity monitoring, has been proposed by Kempf *et al.* [254] in the context of MPLS networks. So far, however, the introduced logical group ports remain unstandardized and are hence not part of shipped OpenFlow switches.

A fundamental question regarding any fast-recovery mechanism concerns the achievable degree of resilience, and in particular, whether it is always possible to find a route under

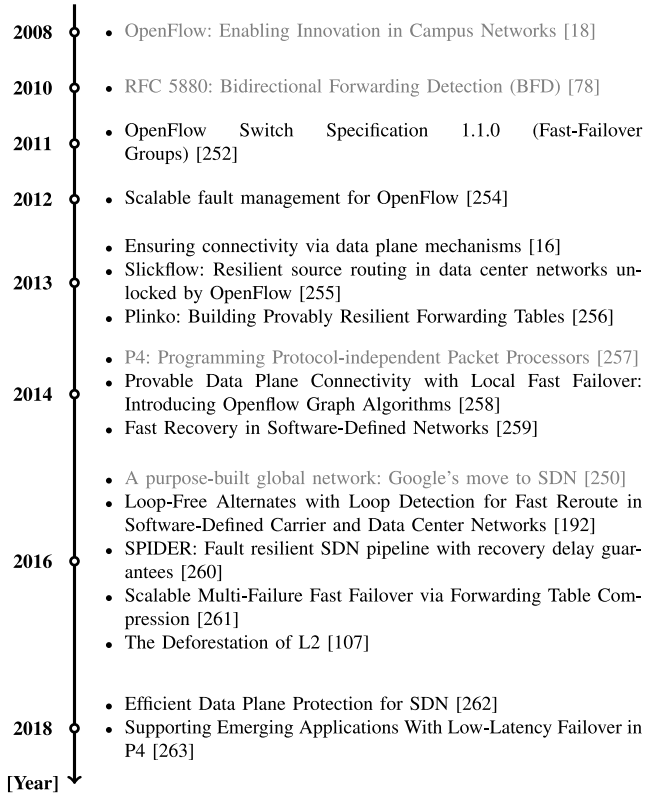

Fig. 28. Timeline of the selected documents and fast-recovery solutions related to programmable networks (entries marked in gray provide additional context).

failures as long as the underlying network is physically connected. For an OpenFlow network, and in a scenario where packet headers can be used to carry information, Borokhovich *et al.* [258] have studied this problem by relying on "graph exploration" FRR mechanisms as explained more in details in the following.

Graph-Exploration FRR: Borokhovich *et al.* [258] showed that such a "perfect resilience" can indeed be implemented. In this approach, the packet carries a counter for each switch, allowing the packet to explore different paths during its traversal through the network. The traversal of the graph can actually be performed in a variety of ways including Depth-First Search (DFS) and Breath-First Search (BFS). This is reminiscent of the Failure-carrying Packets (FCP) [161] approach, used for convergence-free routing (for a detailed discussion of this technique, see Section VIII).

As a case study, we give a detailed **example** for a DFS traversal. Consider the network shown in Fig. 29 with 5 nodes *a*, *b*, *c*, *d*, and *e*, where node *d* is the destination node and *a* is the sender of a packet. The DFS mechanism requires to compute an ordering of all the ports at each switch as shown in the top-left network of Fig. 29. The ordering is used to drive the DFS exploration by trying each outgoing port in the given order until the destination is reached. The first port in the ordering is the default port that is used to send packets in the absence of failures and the set of first ports form a tree rooted at the destination node. The DFS approach requires to store a non-negligible amount of information in the

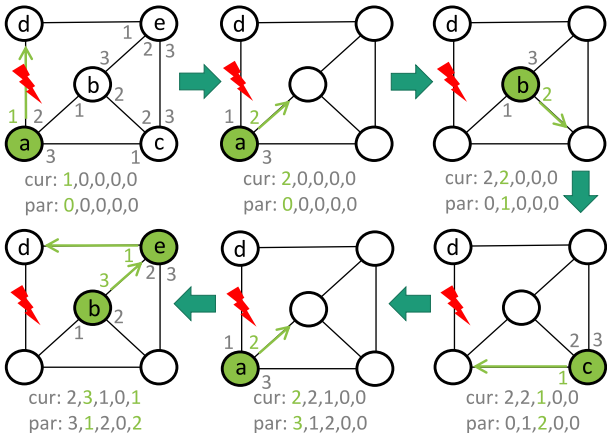

Fig. 29. Depth-first search example.

packets header. Namely, each packet header contains the *currently* explored neighbor at a node (called `cur`) and the *parent* (called `par`) from which a packet has been first received, which are shown below each network in Fig. 29.

Initially, all the current and parent nodes are not initialized, i.e., set to zero. When a node must send a packet, it follows the port ordering starting from the `cur` index of that node. A node always skips a port in the ordering if the port has failed or it is the parent port. When a node has tried all the ports, it sends the packet back to the parent node. This way of forwarding packets is reminiscent of a DFS traversal, which first explores the children nodes and only then it performs backtracking to the parent node. In Fig. 29, node *a* initializes `cur` to 1, which however maps to a failed port. It therefore increments `cur` by 1, i.e., it forwards the packet to the next port in the ordering, which is node *b* (top-center network in the figure). When node *b* receives the packet, it remembers that node *a* is the parent by setting port 1 as the parent. It then increases `cur` from 0 to 1, which however is exactly the parent port so it skips it. The next port leads to node *c*, which in turn sets node *b* (port 2) as the parent port and forwards the packet on port 1 back to node *a* - a potential forwarding loop! When node *a* receives the packet, it first sets node *c* (port 3) as the parent node. It then retrieves from the packet header the last explored port at node *a*, `cur['a']=2`, and increments it to 3, which however is now leading to its parent node so it skips it. Node *a* increments again by 1 in the circular ordering, which leads again to the failed port. Node *a* therefore increments again the current counter and forwards the packet again to node *b* (bottom-center network in the figure). Node *b* retrieves from the packet header the last explored port, `cur['b']=2`, and increments it by 1 to 3. This port now leads to node *e*, which can use its first port to reach the destination.

One disadvantage of the DFS technique (but also for the other graph-exploration techniques) is the packet header overhead, where the packet must remember the state of the currently explored node and parent node for each node in the network. We will see in the next subsection how more programmable paradigms allow to dramatically reduce this overhead without sacrificing the level of resilience achievable by FRR techniques.

We now conclude the OpenFlow subsection by briefly discussing some additional challenges in detecting failures and devising languages for building robust network configurations in OpenFlow. Van Adrichem *et al.* [259] argued that one of the key limitations of achieving a high availability and fast failure detection (below 50 ms) in OpenFlow implementations is that these networks often rely on Ethernet, which in turn relies on relatively infrequent heartbeats. A faster alternative is to use BFD [78], in addition to combining primary and backup paths configured by a central OpenFlow controller.

Another open challenge concerns the design of programming languages for writing fault-tolerant network programs for software-defined networks. The seminal proposal in this context is FatTire [265], a language which is based a new programming construct that allows developers to specify the set of paths that packets may take through the network as well as the degree of fault tolerance required. FatTire's compiler targets the OpenFlow fast-failover mechanism and facilitates simple reasoning about network programs even in the presence of failures.

C. Recovery Concepts in Programmable Data Planes

Lately, programmable data planes [257] emerged which further enrich the capabilities of networks by allowing to deploy customized packet processing algorithms. While several interesting use cases are currently discussed, e.g., related to monitoring or traffic load-balancing, still little is known today about how to implement FRR mechanisms in such systems. In particular, the P4 programming language [257], one of the emerging languages for programming the data plane forwarding behaviour, does not provide built-in support for FRR.

Data-Driven Connectivity (DDC): A seminal approach, ahead of its time, is DDC (for Data-Driven Connectivity) by Liu *et al.* [16], which is motivated by the desire to move the responsibility for connectivity to the data plane. DDC achieve perfect resilience, i.e., packets are forwarded to the correct destination as long as a path exists, similarly to the graph exploration mechanism described for OpenFlow forwarding. It however requires minimal amount of memory to be utilized on the switches. This memory however has to be *transactional* at the per-packet level, i.e., when a packet modifies the content of the memory, the subsequent packet already sees the modification. Transactional memories are today deployed on high-speed programmable switches such as Tofino [266].

At a high level, DDC is influenced by the well-known *link-reversal* algorithm introduced by Gafni and Bertsekas [267], and aims at building a Directed Acyclic Graph (DAG) for each destination. When failures occur, the DAG is recomputed using information gathered from the data-plane packets. DDC hence leaves the network functions that require global knowledge (such as optimizing routes, detecting disconnections, and distributing load) to be handled by the control plane, and moves connectivity maintenance, which has simple yet crucial semantics, to the data plane.

Concretely, forwarding-connectivity is maintained via simple changes in forwarding state predicated only on the destination address and incoming port of an arriving packet, i.e., DDC

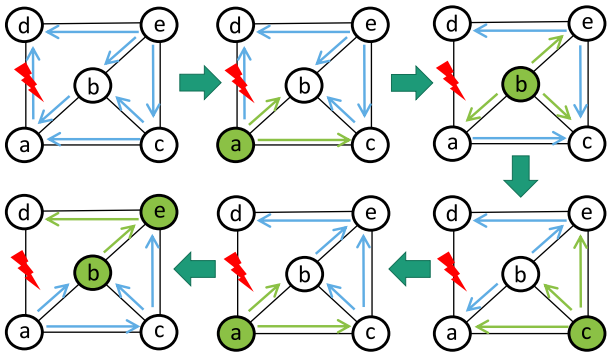

Fig. 30. DDC example.

is a *dynamic* FRR mechanism that needs to modify the forwarding function at the speed of the data-plane. The required state and its corresponding modifications are simple enough to be amenable to data-plane implementations with revised hardware. To implement its service, a DDC node stores three bits of information for each destination and update these based on the incoming packets towards that destination.

We give an **example** of the DDC link-reversal algorithm in Fig. 30. We consider the same network used for the graph-exploration FRR technique explained in Fig. 29 consisting of five nodes. We highlight with blue arrows the directed acyclic routing graph towards the top-left destination node *d*. When the link connecting the bottom-left node *a* to the destination fails, DDC triggers the link-reversal algorithm: if all the link directions are incoming, the direction of all the links are reversed and a packet is sent on any of these outgoing links. The bottom left node adjancent to the failure is the first node to reverse its link directions and forward the packet to its neighbor *b*. Consequently, now *b* has all its links in the "incoming" direction. It therefore reverses their directions and forwards the packet to node *c*. Node *c* also has all its links in the incoming direction so it reverses the link directions and forwards the packet back to node *a* - a potential loop. Node *a* reverses again all its incoming links and chooses *b* to forward its packet. Node *b* now does not have to reverse its links since the link towards node *e* is already in the outgoing direction. This breaks the potential forwarding loop as the packet is now forwarded to node *e* and consequently to its destination *d*. At this point, a new directed acyclic graph spanning all the nodes have been recomputed.

Compared to graph-exploration approaches, DDC achieves identical levels of resiliency, i.e., guaranteed connectivity for any number of failures as long as a physical path exists, but does not incur the exorbitant packet overheads of graph-exploration techniques. DDC only requires to store at each node, for each destination and each port, the current direction of the link plus (1 bit) plus two additional auxiliary bits that are used to synchronize the direction of the link between two nodes in case of multiple link reversal operations.

DDC also describes how to implement some optimizations on the number of reversed links during the reconvergence process so as to speed up reconvergence. In the absence of failures & congestion and assuming an immediate detection of the

failure, none of the packets towards a destination get dropped as long as a physical path exists.

Another approach to design recovery mechanisms in programmable data planes, is to draw from the insights on link layer mechanisms such as AXE [104], [107] (see the related discussion in Section III) which was designed to improve the performance of recovery in Ethernet-based networks. AXE takes a similar approach to DDC by moving the responsibility of maintaining connectivity at the data-plane level.

Programmable data planes can also be used to implement fast-recovery concepts proposed in the context of SDN, but requiring additional features currently not available on OpenFlow switch hardware. For example, *Plinko* [256], [261] requires testing the status of a port in a TCAM match rule, and assuming this feature, achieve a "perfect resiliency": the only reason packets of any flow in a Plinko network will be dropped are congestion, packet corruption, and a partitioning of the network topology. Plinko takes a simple exhaustive approach: in the case of a failure, the switch local to the failure replaces the old route of a packet with a backup route, effectively bouncing the packet around in the network until it either reaches the destination or is dropped because no path exists. When implemented naively, this approach does not scale; however, Plinko can compress multiple forwarding rules into a single TCAM entry, which renders the approach more efficient, allowing Plinko to scale up to ten thousand hosts.

As first recovery concepts for programmable data planes are emerging, researchers also started investigating general primitives to support an efficient recovery. For example, Chiesa *et al.* [263], [268] suggested an FRR primitive which requires just one lookup in a TCAM, and hence outperforms naive implementations as it avoids packet recirculation. This can improve latency and throughput, and, if deployed as a "primitive", can be used together with many existing FRR mechanisms (which is also demonstrated, e.g., for [269]), allowing them to benefit from avoiding packet recirculation.

D. Summary and Discussion

This section provided an overview of the fast-recovery mechanisms provided by programmable networks, and in particular, in OpenFlow and programmable data planes. We have discussed the basic principles underlying these mechanisms, such as group tables, and how the traditional fast failover approaches discussed in the earlier sections can be implemented as well in programmable networks, also pointing out limitations. For example, without additional considerations, a direct implementation of the approaches known from MPLS or IP networks may result in a large number of flow table entries and a suboptimal protection in SDNs; at the same time, programmable data planes enable unprecedented opportunities for more efficient recovery mechanisms, also in terms of latency and throughput.

Fast recovery in programmable networks is the most recent application domain considered in this tutorial, and potentially the most powerful one, given the flexibilities provided in programmable networks. It is hence also likely the domain which still poses the most open research questions. In particular, we only have a very limited understanding of the algorithmic problems underlying the implementation of a highly resilient and efficient failover in programmable networks, e.g., how many failures can be tolerated and how short the resulting failover rules can be kept, e.g., depending on whether and how packet headers can be changed. These algorithmic problems are further complicated by the specific technology and data structures that are used to implement recovery in programmable networks, e.g., related to recirculation overheads. Another important issue concerns the development of intuitive high-level concepts for the network programming language itself.

VIII. Technology-Agnostic Fast-Recovery Methods

There exists a number of interesting FRR mechanisms which do not exactly fit any of the specific layers and technologies discussed above.

A. Motivation and Background

In this section, we introduce the reader to some selected solutions which are based on fundamental techniques one should be familiar with. These solutions have sometimes been proposed in specific contexts and layers discussed above, but require non-trivial additional features, which motivates us to study them here. Some of the recovery concepts discussed below are also fundamental and proposed independently from a specific technology. We believe that these concepts are hence also particularly interesting when planning the next generation of reliable communication technologies, providing additional features for fast recovery.

B. General Recovery Concepts

1) Rerouting Along Arborescences: In order to achieve a very high degree of resilience, several previous works [163]–[165], [228] introduced an algorithmic approach based on the idea of covering the network with arc-disjoint directed *arborescences* rooted at the destination. Network decompositions into arc-disjoint arborescences can be computed in polynomial time, and enable a simple forwarding scheme: when encountering a link failure along one arborescence, a packet can simply follow an alternative arborescence. This approach comes in different flavors, such as deterministic [165] and randomized [164], or without [164] and with [165] packet header rewriting. In general, the approach requires *input port matching* to distinguish on which arborescence a packet is being routed. This however is practical, since many routers maintain a routing table at each line card of the interface for look-up efficiency. In the most simple case, a network can be decomposed into arc-disjoint Hamilton cycles. For example, Fig. 31 shows an example for the case of 2-dimensional torus graphs. In this case, the solution is known to achieve the maximum robustness: a network of degree k can tolerate up to $k - 1$ failures, without losing connectivity. A more general example is given in Fig. 32: here, arborescences rooted at

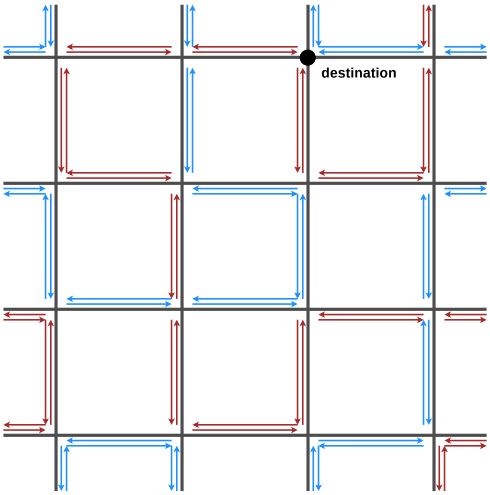

Fig. 31. Decomposition of a 2-dimensional torus into arc-disjoint Hamilton cycles.

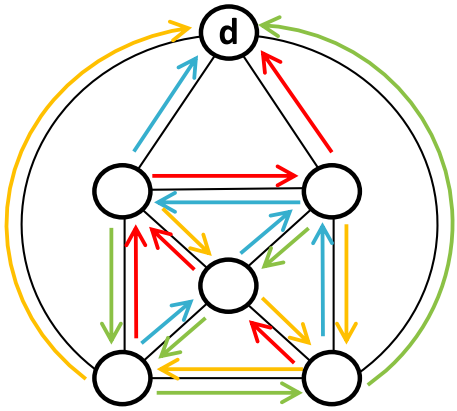

Fig. 32. Decomposition of a general graph into arc-disjoint arborescences.

the destination can be of higher degree. While such decompositions always exist, the challenge introduced in the general setting regards the order in which alternative arborescences are tried during failover. Interestingly, today, it is still an open question whether the maximum robustness can be achieved in the general case as well. However, Chiesa *et al.* showed in [163] that generally, the approach can tolerate at least half of the maximally possible link failures. They also showed that a resilience to $k-1$ link failures can be tolerated using 3 bits in the packets header or creating $r-1<k$ copies of a packet, where r is the number of failed links hit by a packet. While these arborescence-based approaches provide a high resilience, one disadvantage of the approach is that it can lead to long failover paths (consider again the torus example in Fig. 31), introducing latency and load. The latter has been addressed in a line of work by Pignolet *et al.* [270], [271], e.g., relying on combinatorial designs or postprocessing of the arborescences [212], [272], [273].

2) Techniques Beyond Arborescences: The use of spanning trees, directed acyclic graphs, or arborescences for resilient routing, may come with the limitation that they only select

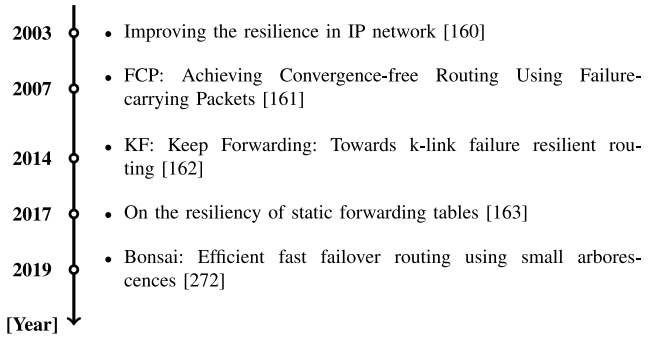

Fig. 33. Timeline of the selected documents and solutions related to technology-agnostic fast-recovery methods.

links from such subgraphs. An interesting more general approach is *Keep Forwarding* (*KF*) [162]. KF is based on a "Partial Structural Network (PSN)": the idea is to utilize *all* links and only determine link directions for a *subset* of links to improve resilience (somehow similar to DDC [16] which we discussed in Section VII-C). As such, KF can handle multiple failures with only small path stretch, and does not require packet labeling or state recording. To achieve these properties, KF needs to be "input-port aware": forwarding depends not only on the destination but also on the input port. The authors also derive a *Graph Imperfectness Theorem*, showing that for an arbitrary graph if any component of it is "imperfect", there will be no static rule-based routing guaranteeing the "perfect" resilience, even if the graph remains connected after failures. Nevertheless, the authors show in experiments that KF provides a high resilience.

A higher resilience can be achieved by methods which exploit header rewriting, such as Failure-Carrying Packets (FCP) [161]: in FCP, packets can autonomously discover a working path without requiring completely up-to-date state in routers. FCP takes advantage of the fact that typically a network topology does not undergo arbitrary changes, but there is a well-defined set of "potential links" that does not change very often: while the set of the potential links that are *actually functioning* at any particular time can fluctuate, e.g., depending on link failures and repairs, the set of *potential* links is governed by much slower processes (i.e., decommissioning a link, installing a link, negotiating a peering relationship). Thus, one can use fairly standard techniques to give all routers a consistent view of the potential set of links, which is called the Network Map. Motivated by this observation, FCP adopts a link-state approach in that every router has a consistent network map. Since all routers have the same network map, packets only need to carry information about which of these links have failed at the current instant. This "failure-carrying packets approach" ensures that when a packet arrives at a router, that router knows about any relevant failures on the packet's previous path, and can hence determine the shortest remaining path to the destination (which may also be precomputed in the data plane). This eliminates the need for the routing protocol to immediately propagate failure information to all routers, yet allows packets to be routed around failed links in a consistent loop-free manner. There

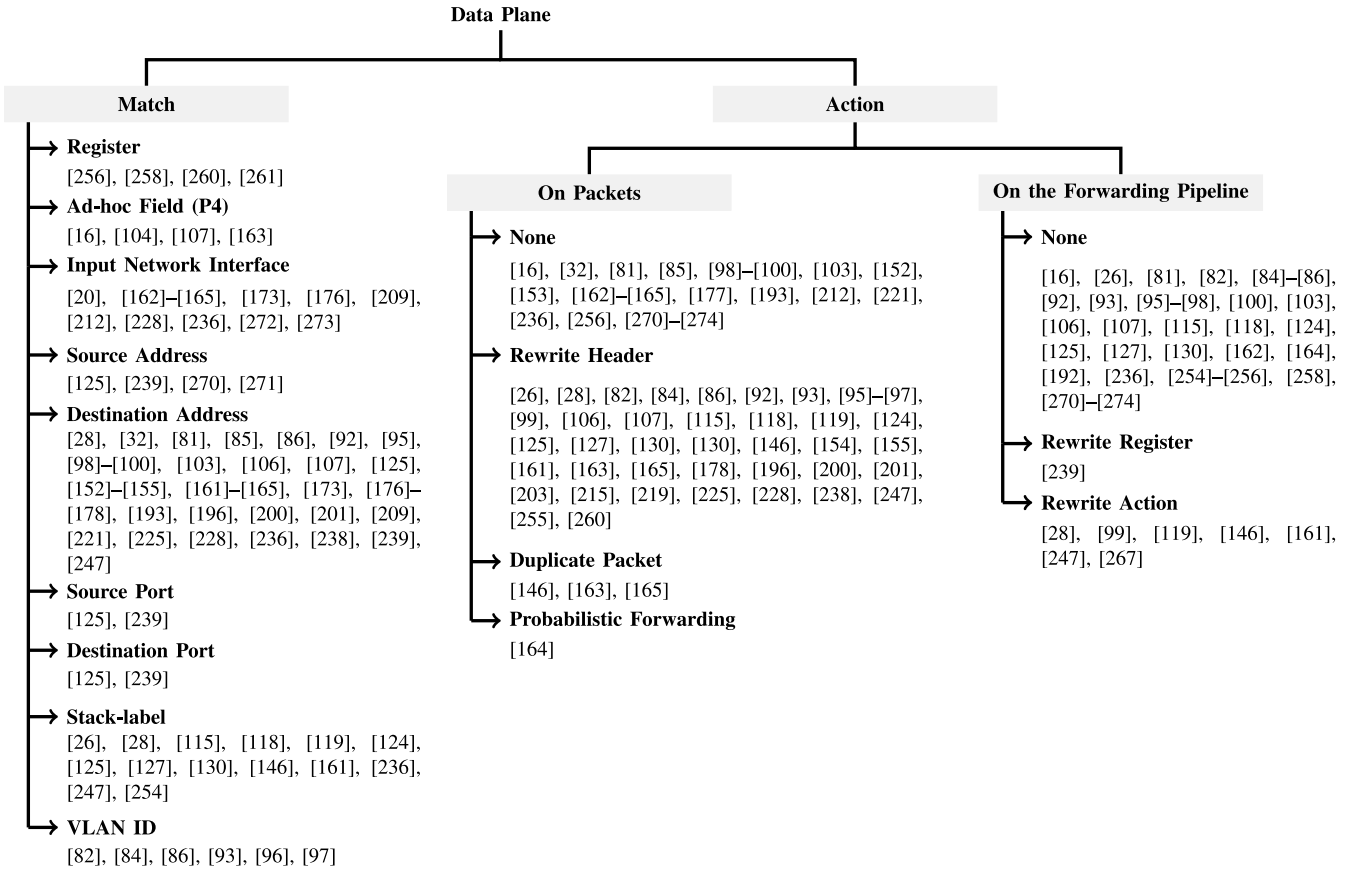

Fig. 34. Classification of the presented fast-recovery mechanisms with respect to the match-action operations performed in the data plane.

is also a Source-Routing FCP variant that provides similar properties even if the network maps are inconsistent, at the expense of additional overhead in packet headers. More concretely, in the source-routing variant, like in the basic FCP, a node adds the failed link to the packet header, but replaces the source route in the header with a newly computed route, if any exists, to the destination.

There exist several additional interesting technology-agnostic approaches. To just give one more example: An elegant solution to provide a certain degree of resilience is described in the O2 ("out-degree 2") paper [160]: in order to reduce the outage time compared to traditional IP networks, O2 provides each node with at least two disjoint next hops towards any given destination, allowing each node to locally and quickly re-distribute the traffic to the remaining next hop(s) if a route fails. The resulting paths are loop-free but may increase link loads slightly, due to increased path lengths.

3) Topological Support for Fast Recovery: Fast rerouting can also be supported at the network topology level, which introduces an interesting additional dimension to the fast-recovery space. Likely, a more redundant topology can tolerate more link failures; however, while this is obvious for centralized routing algorithms, it is not necessarily clear how to exploit redundancy in local fast rerouting algorithms. An interesting approach in this context is F10 [269]: a novel network topology reminiscent of traditional fat trees, with better fault recovery properties. In particular, by a clever re-wiring, F10 topologies support the design of failover protocols which can reestablish connectivity and load balancing more generally and quickly.

C. Summary and Discussion

We discussed several fast-recovery methods which are independent of technologies, from failure-carrying packets over arborescence-based approaches (which do not require packet header modifications), to the design of alternative topologies which explicitly account for the fast failover performance.

There are several interesting open research issues. In particular, the question of how much information needs to be carried in packets to achieve certain resilience guarantees remains an open problem. For example, it is not clear whether the capability of input port matching is sufficient to guarantee connectivity in a k-connected graph under up to $k-1$ failures, using arborescence-based approaches or in general. Another interesting open problem regards the design of network topologies for efficient fast rerouting beyond fat-trees.

IX. Classification Summary

Given our understanding of the various recovery concepts which are used by the different technologies and which are often reoccurring on the different layers, we now take a step back and identify and classify the principles underlying the specific approaches discussed above.

A most simple form of fast recovery that we have already encountered multiple times is based on *static* failover routing:

the forwarding behavior is statically pre-defined, e.g., using conditional per-node failover rules which only apply in case of failures, and cannot be changed during the failover. For example, it is not possible to implement link reversal algorithms (e.g., [267]), as they require dynamic routing at the nodes.

Considering that a variety of parameters may influence the decision about the preferred **alternative output interface**, fast-recovery mechanisms can come in different flavors (see the overall classification shown in Fig. 34), for example:

- *Beyond Destination Address Matching:* Can forwarding depend on packet header fields other than the destination address? Solutions based solely on the destination address such as [164] are attractive, as they may require less forwarding rules. More general solutions which, e.g., also depend on the source address [270] as well as the source and destination port numbers used by transport-layer protocols [125], [239], may enable a more fine-grained traffic engineering scheme and thus reduce network load during the failover.
- *With or Without Input Network Interface Matching:* Can the forwarding action to be applied to a packet depend on the incoming link on which it arrived? Input interface matching can improve the resilience and quality of fast rerouting (in particular, by detecting and avoiding forwarding loops) [163], [212], but may render the forwarding logic more complex.
- *Stack-Label Matching:* Can messages be forwarded based on the label currently occupying the top position in a stack embedded in the message header? Stack-label matching enables flexible forwarding along pre-established paths in the network, without performing additional routing table lookups based on the values of the primary fields describing the source and the destination of the message [26], [116], [117]. Subsequent detours may be initiated simply by pushing a different label on the stack when necessary. At the same time, stack-label matching requires that the involved devices support the related extensions as well as a label distribution protocol maintaining consistency of the mapping between labels and the corresponding paths.
- *VLAN identifier matching:* Can the forwarding decision depend on the VLAN identifier stored in the message header? Unless the limited range of allowed values is likely to become an issue in specific deployments (especially those involving legacy network devices), VLAN identifiers might be used for fast-recovery purposes as a convenient signaling channel leveraging the widely-supported network standard. Whenever a failure is encountered, the local Ethernet switch would typically set the VLAN identifier to a different value associated with one of the alternative spanning trees and then it would forward the message along the selected tree. Downstream switches would forward the message along the same tree until the destination is reached, or until another failed link is detected on the intended path towards the destination. Note that this method may not be used together with the typical VLAN functionality, as it would allow messages assigned to one VLAN to leak into a different VLAN, bypassing the security policy defined on routers.
- *Register/Ad-hoc Field Matching:* Can programmable network devices make forwarding decisions based on additional sources of information, for example, values stored in hardware registers or in ad-hoc fields associated with the processed packets? Considering an increasing range of potential applications involving programmable network devices as well as substantial efforts supporting the development of future self-driving networks, forwarding decisions may also be influenced by external factors represented by the current values of internal registers and ad-hoc fields. Consequently, the flexibility offered by the underlying systems may lead to a better integration with specific environments and to the development of unique fast-recovery solutions. The related advantage is that custom-designed packet processing pipelines may be evaluated and deployed much faster and potentially at a lower cost, compared to the equivalent proprietary off-the-shelf solutions. At the same time, new designs will remain constrained by the limitations of programmable devices and by increasing performance requirements. A good example illustrating the trade-off between the resource utilization and performance (in terms of latency and throughput) has been presented in [268].

Based on the specific subset of parameters which are used by an algorithm to determine the preferred alternative output network interface, the corresponding **actions** may also be triggered if necessary, both in the context of packets as well as the entire forwarding pipelines maintained by network devices (see the corresponding branch in Fig. 34), for example:

- *Packet-Header Rewriting:* Can nodes rewrite packet headers depending on failed links, e.g., to store information about the failures encountered by a packet along its route? Packet-header rewriting is known to simplify the design of highly resilient fast-recovery algorithms [165], [258]. A well-known example are fast-reroute mechanisms based on failure-carrying packets [161]. Rewriting may also be performed on other objects, such as the internal registers used by programmable devices.
- *Action Rewriting:* Can nodes rewrite the intended action assigned to subsequent matching messages, based on the detected signs of failure? To reduce the negative consequences (such as forwarding loops) further, or to optimize the resource utilization in the network, programmable devices may change the preferred action performed on the matching messages. However, an important related concern is preserving network stability following the failure.

Existing mechanisms also differ in terms of their **objectives**, such as:

- *Connectivity:* The most basic and most studied objective is to preserve connectivity. Ideally, a fast-recovery mechanism should always find a route to the destination, as long as the network remains physically connected. Especially the design of mechanisms without header rewriting has received much attention [20], [164], [165], [228], [256], [274], [275].

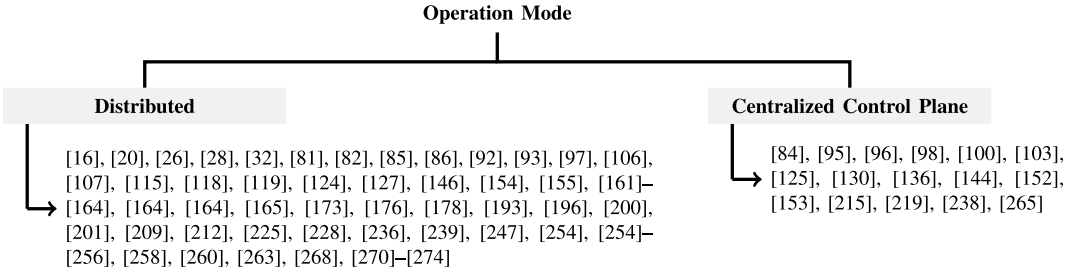

Fig. 35. Classification of the presented fast-recovery mechanisms with respect to their operation mode.

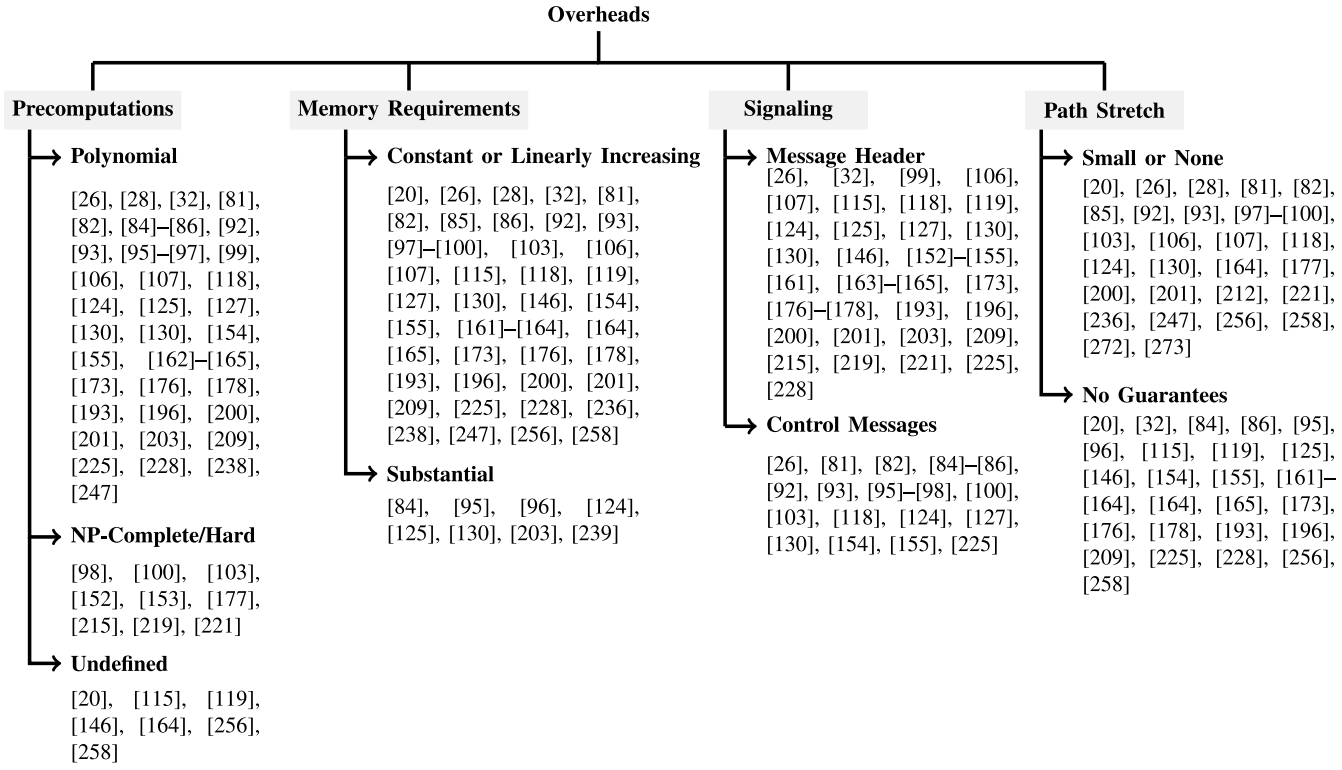

Fig. 36. Classification of the presented fast-recovery mechanisms with respect to the implementation- and operation-related overheads.

- *Load:* Another important objective is to avoid network congestion during and after failover, see, e.g., [270], [276].

Additionally, they may also differ in terms of their **operation mode**, defined here as either distributed or centralized operation (see Fig. 35).

- *Distributed Operation:* The majority of the solutions discussed in this paper have been designed to operate in a distributed fashion. The key advantage of this approach is that network devices can collect the necessary information, develop their internal state, and prepare for future failures without relying on the other devices in the network. At the same time, the recovery decisions may not always be optimal in the case of multiple failed elements, as the involved devices do not coordinate their response with each other.
- *Centralized Operation:* Centralized fast-recovery approaches are still expected to be able to make local decisions without significant delay. However, in this scenario, forwarding devices usually depend on a central unit with respect to other key tasks, such as precomputation of the preferred (if necessary, optimal) network-wide recovery strategy taking into account additional performance-critical factors (e.g., network load). As the centralized unit may need to collect the necessary information from the entire network or domain, process it, and then update the fast-recovery rules on forwarding devices, it may quickly become the bottleneck. Further, as the centralized unit and its connections may also fail or be subject to targeted attacks, the capability of the network to respond effectively to subsequent failures might become severely limited in such cases, unless additional measures are deployed to counteract this issue.

Deployment of fast-recovery solutions is always associated with implementation- and operation-related **overheads** (see Fig. 36), for example.

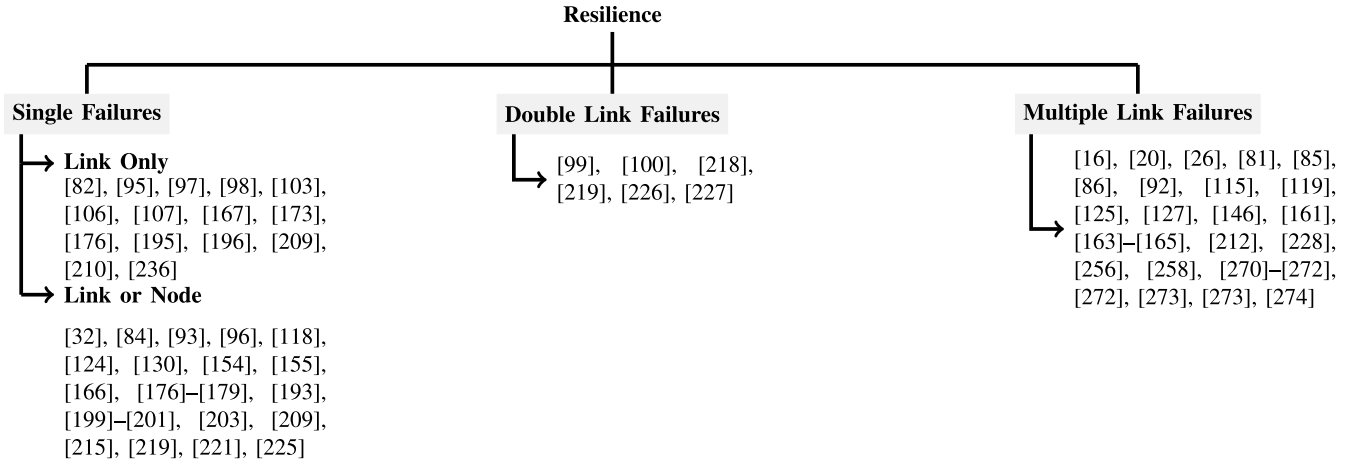

Fig. 37. Classification of the presented fast-recovery mechanisms with respect to the maximum number of failures they were designed to deal with. Note that under some specific circumstances, fast-recovery mechanisms designed to handle single failures might still be able to deal with multiple failures effectively (for example, when the failed components are located in different regions of a well-connected network).

- *Precomputations:* The internal *knowledge* of the preferred recovery actions often results from precomputations which may be performed either locally or by an external unit (in the case of the centralized operation mode). It needs to be emphasized that this process is initiated during the normal network operation period and in most cases should have completed by the time of the next failure event. Consequently, at the time of failure, the involved forwarding devices can redirect messages almost instantly. The related cost depends on the algorithmic complexity of the design, ranging from polynomial-time solutions to NP-hard problems.
- *Memory Requirements:* To be able to perform the failover within milliseconds, forwarding devices need to develop and maintain internal state information that will define the preferred alternative network interfaces in different failure scenarios. In the context of modern programmable switches relying on expensive Ternary Content-Addressable Memory (TCAM) modules that are able to perform wildcard matches, it is desired that only the minimum required amount of information be stored in such memory modules to preserve space. Depending on how particular fast-recovery solutions have been designed, the memory-related overhead may either be constant or be bound to some parameters of the network such as number of destination prefixes.
- *Signaling:* Some fast-recovery designs rely on additional signaling to carry important information between the involved nodes. In particular, the selected bits of a message header may be used to indicate the preferred path in the network between the source and destination nodes. Alternatively, if the expected modifications of the message header would disrupt the operation of other network protocols and mechanisms such as VLANs, dedicated control messages may be exchanged by forwarding devices rather than a common protocol. Depending on the specific method in use, the related cost may result from the data overhead or from the additional operation limitations.
- *Incurred Path Stretch:* Many existing fast-recovery solutions cannot deal with simultaneous failures of multiple network elements effectively. Even those having such a capability may still be unable to forward messages around the failed components along the shortest possible paths. Consequently, they may not provide any guarantees regarding the observed path length (or stretch). However, the recent advancements in static fast-recovery mechanisms relying on modified arborescence-based network decompositions provide one of the possible solutions to this issue [212], [272].

Finally, the effectiveness of existing fast-recovery mechanisms in terms of the maximum offered **resilience capabilities** is also diverse (see Fig. 37).

- *Single Failures:* Early fast-recovery solutions, especially those operating in the link layer, have been designed to respond to single link or node failures. On one hand, simultaneous failures of two or more network elements are less likely to happen than a failure of a single component [139], which means that being able to restore connectivity just in the case of single failures already covers the most frequent scenario. At the same time, in large networks involving numerous different devices and subsystems undergoing regular maintenance activities, such events are not uncommon and single-failure-recovery strategies may not always be successful, leading to increased packet losses, disruptive delay, and even persistent or transient forwarding loops. Indeed, according to [139], scheduled maintenance activities alone may have caused 20% of all observed failures in an operational IP backbone network, while almost 30% of all unplanned failures affected multiple links at a time. In this context, it needs to be emphasized that both the resilience requirements as well as the overall complexity of networked systems have been constantly increasing over time.

- *Double Link Failures:* To extend the fast-recovery capabilities of computer and communication networks beyond the single-failure scenario, improved designs were developed that were able to deal with double link failures. Considering the evolution of fast-recovery strategies, this was an intermediate step towards more general strategies handling simultaneous failures of multiple network components.
- *Multiple Link Failures:* Dealing with multiple link failures effectively depends not only on the design of a recovery mechanism, but also on the physical network topology. One of the key related parameters coming from graph theory is the edge connectivity of the network topology, further referred to as k. In particular, if up to $k-1$ links in a given k-connected network suddenly become unavailable, the remaining links can still be used to reach any destination in the network.[10] An example group of the related recent solutions is focused on static fast-reroute mechanisms. The underlying **algorithmic techniques** include.
  - *Arborescence-Based Network Decompositions:* By decomposing the network into a set of arc-disjoint arborescences which are rooted at the destination [163], [164] (such a decomposition can be computed in polynomial time), a high degree of resilience can be achieved: when encountering a link failure along one arborescence, a packet can simply follow an alternative arborescence.
  - *Combinatorial Block Designs:* Pignolet *et al.* [270], [276] observed and exploited a connection of fast rerouting problems to combinatorial block designs: static resilient routing boils down a subfield in distributed computing which does not allow for communication.

  Ideally, a static fast rerouting algorithm ensures connectivity whenever the underlying physical network is connected. Feigenbaum *et al.* [20] showed that without packet header rewriting, such an ideal static resilience cannot be achieved. A weaker notion of resilience was introduced by Chiesa *et al.* [163], [164]: the authors showed that there exist randomized static rerouting algorithms which tolerate $k-1$ link failures if the underlying network is k-edge connected, even without header rewriting. At the same time, a fundamental open problem is whether for any k-connected graph, one can find deterministic failover routing functions that are robust to any $k-1$ failures.

X. Discussion

While our main focus is on data plane fast-recovery mechanisms, we point out that additional challenges may arise from the interaction between the control plane and the data plane during recovery. In particular, data plane mechanisms are often seen as a "first line of defense": a way to react *quickly*, but possibly *suboptimally*, to failures. For instance, a fast-recovery path may allow to bypass a failed network component instantaneously but, by directing additional traffic to links that in normal circumstances would not have to cope with that traffic, it may cause unexpected congestion and packet loss. Further service disruptions may occur due routing transients, like short-lived forwarding loops (so called microloops) and routing blackholes, caused by the inconsistent timing of updating the data-plane configuration at different routers. Worse yet, such phenomena may spread to remote parts of the network, harming flows that would not have been affected by the failure otherwise.

Routing transients lead to minor performance degradation as long as they are short-lived, lasting not much longer than the full *restoration completion time* (recall the recovery procedure timeline in Fig. 7, Section II) and disappearing before the network would enter the *normalization phase*. Typical examples are a router reboot event due to a software bug or a flapping interface going through an up-down-up cycle; these events usually last only a couple of milliseconds or seconds at the worst. If the failure proves long-lived so that the normalization phase is initiated (recall again Fig. 7, Section II), then in the second stage the control plane must reconverge to a new static route allocation that is optimized for the changed network configuration, with the failed component permanently removed from the topology. Finally, another transient phase takes place after the normalization process terminates and the failed component comes back online.

In order to avoid performance degradation during routing transients, there is a need to carefully orchestrate the way the control plane interacts with the data plane, as we discuss next. First, in Section X-A we sketch several schemes from the literature to schedule dataplane updates across a network domain to avoid routing transients. Then, in Section X-B we survey proposals for traffic-engineering the recovery paths in order to eliminate transient congestion.

A. Reconvergence and Transient Loops

Different approaches in the literature deal with the interaction between data-plane fast reroute and the control-plane reconvergence process in different ways. We discuss some examples in the context of intra-domain and inter-domain routing, and programmable networks.

It is perhaps the context of *shortest-path-based distributed intra-domain routing* where routing transients manifest their adverse effect most visibly. This is on the one hand due to the inherent distributed nature of IGPs, where there is minimal or no central oversight on when and how data-plane updates are performed, and on the other hand because of the fundamental properties of shortest-path routing that allow for two routers, e.g., one aware of a failure and another one that is not, to appoint each other as the next-hop towards a particular IP destination prefix (leading to a microloop) or failing to synchronize at a consistent forwarding path (leading to a transient or permanent blackhole).

For an in-depth covering on the timing aspects of link-state IGP reconvergence after a network topology change, the reader

[10]Note that this condition is only related to graph connectivity, while in real networked environments, several additional factors need to be considered as well, such as traffic characteristics, the load of particular network components, and the relevant traffic engineering policies.

is referred to [15]. The paper also proposes the use of incremental SPF algorithms and FIB updates to make the process faster. Note that the IGP convergence time can be improved even further by careful tuning, as long as it does not affect the stability of the network considerably [15], [277].

As it turns out, IGP tuning alone is not sufficient to completely eliminate all IGP routing transients. A cornerstone contribution to reach this end is made in [278]. The paper proposes a mechanism to control OSPF administrative link weights for adding links to, or removing links from, the network, by progressively changing the shortest-path link weight metric on certain "key" links, ensuring that in each step the topology change occurs in a loop-free manner. Note that this mechanism needs a central entity to plan and drive the process; hence, it is best used for a non-urgent (management action) link or node shutdowns and metric changes.

The Ordered FIB update (oFIB) proposal, reaching the Informational RFC status in 2013, brings this work further [279]. The idea is to sequence the FIB updates by each router computing a rank that defines the time at which it can safely update its FIB. This approach can be used to administratively add or remove links (similarly to [278]), when there is sufficient time to pace out FIB updates, but it is also useful in conjunction with a fast-reroute mechanism when a link or node failure persists and the task is to re-converge the transient routes created by FRR to the static IGP shortest-path routes.

Due to the specifics of BGP, *inter-domain fast IP recovery* requires different techniques. During BGP reconvergence, BGP withdrawals may be sent from any AS adjacent to a link failure to all its neighbors. These neighbors may, in response to these withdrawals, stop forwarding traffic via said ASes and, consequently, drop traffic. This is undesirable especially in the case when there still exists a fast-recovery path that does not traverse the failed link. R-BGP, an FRR extension to BGP, uses two simple ideas to avoid this [236]. First, a node withdraws a route to a neighboring AS only when it is sure it will not offer it a "valley-free" route in the post-convergence state. Second, an AS keeps forwarding traffic on a withdrawn route until it either receives a new BGP route that does not traverse the failed link or it understands it will not receive a valid BGP route in the post-convergence state. We refer the reader to the original paper [236] for the details.

In the context of *software-defined networks*, different techniques are required for a smooth control-plane–data-plane interaction during FRR. Even if the control plane is centralized (as it is the case in the standard SDN setup) the data plane is not, and therefore it is essential for the former to carefully orchestrate the update of the latter. This is a surprisingly complex problem that has received much attention in the research community lately; below we only sketch a couple of examples from the SDN literature and we refer the reader to the recent survey [280] for an in-depth coverage. For instance, in DDC (see Section VII-C, [16]), fast data-plane re-convergence at any possible transient stage is carefully reconciled with the SDN centralized control plane in a way as to eliminate microloops during the transition. An orthogonal approach is taken in [281], where the data-plane-based FRR mechanism is used to obtain additional failure information for the SDN control plane for better failover planning [239].

Control-plane–data-plane interaction becomes even more problematic when *distributed and a centralized control planes coexist* in the same network, each modifying the FIBs of the same set of network switches without synchronizing. Recently, Vissicchio *et al.* [282] developed a general theory that characterizes the kinds of routing and forwarding anomalies that may occur if the two (or more) control planes act independently from each other.

B. Traffic Engineering and Fast Reconvergence

As discussed above, careful preparation is needed during the transients after a failure event to avoid that certain links, or entire network regions, become overwhelmed with traffic bypassing the failure and preclude cascading failures due to the resultant congestion. Accomplishing this traffic optimization task requires a concerted effort on the part of the data plane and the control plane, so that the recovery paths fulfill the traffic engineering goals even under failures. There is a breadth of standards, operational best practices, and research proposals to reach this goal; below we give a non-exhaustive overview of some of the most typical approaches; for more detail, refer to the original papers and the related surveys [280].

A significant number of fast-recovery mechanisms do not natively support fine-grained traffic engineering during recovery. In these cases, recovery occurs on a *best-effort* basis, merely hoping that the failover paths will be "good enough" in that there is enough over-provisioned spare capacity available in the network to avoid congestion during fast recovery. This approach is adopted most often for data-plane technologies that otherwise provide very little in the way of optimizing forwarding paths, like L2 protocols (e.g., AXE [104], [107], see Section III), intra-domain IP fast reroute (e.g., the original LFA proposal [32], see Section V) or inter-domain IP routing where BGP does not provide fine-grained mechanisms for optimizing AS-AS paths (e.g., [236], see Section VI).

The second approach is to compute recovery paths *on-demand*, in a way to ensure that the recovery paths will have enough capacity to handle failover traffic. This approach is used for data-plane technologies where the control plane can surgically optimize the forwarding and recovery paths with respect to arbitrary traffic engineering goals. In this case, the selected TE designs could be implemented on top of the default fast-recovery scheme, overriding the decision made by the underlying fast-recovery mechanism based on additional TE metrics (provided that multiple backup network interfaces are available). In the context of MPLS FRR (see Section IV), the functional capabilities for reconciling traffic engineering policies in FRR have been defined in [114]. The related *resilience attributes* may either be limited to indicate just which recovery procedure should be applied to failed traffic trunks (*basic* resilience attribute) or they can also specify detailed actions that should be taken in the case of failure (*extended* resilience attribute). In particular, the extended resilience attribute may define a set of backup paths as well

as the rules which control the relative preference of each path. To be able to impose traffic engineering policies, MPLS relies on close interaction with routing. Further extensions to MPLS provide a way to define label-switched tunnels that can be automatically routed away from failed network components, congested links, and bottlenecks [117], and the additional extensions proposed in [26] introduce the capability of MPLS to establish backup LSP tunnels in advance; see, e.g., [137] for a linear TE-aware optimization model for link, node, and bandwidth protection in the context of MPLS.

The third approach encompasses a broad set of models, methodologies and algorithms for the *co-design* of the default, failure-free forwarding paths and the failover paths. In intra-domain shortest-path-based IP routing any non-trivial change in the IGP link weights applied to bypass a failed network component will necessarily reroute some, or even all, traffic instances that would otherwise not be affected by the failure [278], [279]. To prevent such cascading service disruptions, [283, Sec. 4] presents a set of local search algorithms to co-design the default IGP weights and the failover IGP weights so that the number of necessary weight changes, and hence the number of traffic instances being rerouted, is minimal. They show that in many cases as few as 3 weight changes is enough to attain "good enough" performance. In networks where a centralized control plane is available, protection routing [152], [153] provides several optimal and heuristic algorithms for traffic engineering failover paths, and R3 [125] shows an elegant extension of oblivious routing, converting "traffic uncertainty" to "topology uncertainty due to an unexpected failure case" to reach the same goal; refer to Section V and Section IV for a detailed overview.

The general observation is that *survivability and performance are fundamentally at odds* [152], [153], by the need to stack valuable transmission capacity to accommodate bypass paths that could otherwise be used to increase failure-free throughput. As it turns out, however, *simple algorithmic techniques can generally find good trade-offs* to reconcile these two conflicting requisites, but to achieve this a careful control-plane–data-plane co-design approach is necessary.

XI. Lessons Learned and Future Directions

This survey covers a rich body of literature of fast data-plane recovery mechanisms for packet-switched networks, ranging from traditional layer-2 network technologies up to today's programmable network protocols and paradigms. In this section, we aim at summarizing the main lessons learned in the design of fast-recovery mechanisms across the different technology layers and draw a few concrete future directions.

The Reaction Time Highly Depends on the Operational Layer: Layer-2 networks require fast failover operations that operate at smaller time granularities than what one would require at the Internet level. As discussed in Section VI, today's convergence of the inter-domain BGP routing protocol is in the order of tens of seconds or even minutes. In contrast, intra-domain IP or Layer-2 networks require failover mechanisms to operate in the sub 50ms time scale for today's needs. At the core of the difference is the inherent difficulty to orchestrate a failover recovery in the Internet where control is distributed, visibility is limited, technologies heterogenous, and interests among domains not necessarily aligning. Within singly administered domains, optimized algorithms and mechanisms can instead be devised to achieve faster reaction time.

The Forwarding Protocols Affect the Deployable Techniques: When looking at a single domain, one critical aspect to take into account is the forwarding properties of the protocols used to move traffic within a network. Consider an Ethernet Layer-2 network and an IP network. In the first case, the lack of a Time-To-Live field in the packet headers require Layer-2 failover techniques to avoid generating any possible transient forwarding loops, as these would inevitably congest and break the network, especially if any packet is flooded. In the case of IP networks, transient forwarding loops are detrimental for performance. They do not necessarily break the network, however, as routers eventually drop packets forwarded in a loop.

Hardware Support and Programmability are Crucial: Fast failover mechanisms have evolved dramatically across over the last four decades. Adapting to the emerging technologies, more advanced fast failover mechanisms have been designed. Traditional Layer-2 Ethernet networks relied on spanning tree protocols with low congerence due to the need to avoid transient loops. Rapid spanning tree protocols mitigate some of these convergence issues but require support from the underlying devices. The ultimate decision on which fast failover mechanisms should be deployed in a network often boils down to understanding the heterogeneity of the network devices in terms of both vendors and switch/router generations. For instance, we note that while most of the major vendors support today IP LFA and remote LFA, a variety of them may not support them or only support software implementations [284] As for OpenFlow devices, it is only from the OpenFlow 1.3 or above specification that fast-reroute groups have been optionally introduced and deployed by the different vendors. Recently, more programmable network devices, such as P4 switches, have made the deployment of different types of FRR mechanisms easier by decoupling the specific forwarding capabilities of a network device from the FRR mechanisms supported by it. Today's network programmers can compile different types of FRR mechanisms onto P4 switches (e.g., DDC [285]) and adapt the algorithms to their needs without the need to wait for patches from the network vendors and/or chip manufactures.

Legacy-Compatibility Eases Deployment of new FRR: Operators wish to improve the resilience of a network to failures without replacing all the network devices but rather upgrading just a *few* of their devices. Local fast-reroute techniques such as LFA are a good example of such incrementally deployable technologies where critical routers could be upgraded to support LFA during a network deployment transition. Today, we observe that Segment Routing (SR) technologies offer similar advantages when deployed with IPv6. Since FRR policies are encoded in the general IPv6 segment options, operators do not need to upgrade routers in their

network that do not need to interpret these segments, lowering any barrier for deployment.

Enhanced Communication Models are Poorly Supported in FRR: The vast majority of fast-recovery mechanisms in the literature address the basic setting: single-layer, unicast communication. This is not surprising: handling a failure is simplest when the information to be learned, and the actions to be performed, during recovery is constrained into a single transmission technology layer, and concerns point-to-point communication only. However, communication networks are inherently multi-layer, in that one transmission technology, such as Optical Transport Network, serves as a carrier for another transmission protocol such as IP [42]. In this setting, it becomes notoriously difficult to identify the exact layer in the stack where a failure occurs [47] and fix the "owner", i.e., the layer that should handle it. While several inter-layer cooperative fast-recovery designs are already available (e.g., bottom-up, top-down, and integrated strategies, see [73]), the general setting is for further research. Similarly, although the unicast setting is excessively addressed in the literature and initial designs are available for fast multicast recovery as well, like an 1+1 [286] and an 1:1 protection scheme [287] for BIER (Bit Index Explicit Replication, [288]), fast recovery for the general multicast, anycast, and geocast communication mode remains a challenging open problem for now.

The Future of Fast-Recovery Mechanisms: Programmable networks offer great opportunities in terms of being able to design, deploy, and experiment with different fast-recovery data-plane mechanisms in a network. Given the emerging flexibilities of software-defined networks, one may wonder whether there are some methods which are preferable over others. In general, we can conclude that there is no free lunch. For example, there can be trade-offs between resilience and efficiency: in scenarios where it is sufficient to provide resilience against one failure, it may be preferable to choose simple solutions, as state-of-the-art mechanisms for many failures can lead to long routes, even under a small number of failures (e.g., approaches based on arborescences). We have also seen that the ability to change the header depending on the failures can greatly improve resilience: it may be possible to achieve a perfect resilience, i.e., to stay connected as long as the underlying network is connected; this is known to be impossible without header rewriting. An example where a "technology detail" can make a big difference regards the ability to match the input port: destination-based-only routing is often less powerful (e.g., if headers are immutable) than routing which can also depend on the source, and if additionally the input port can be matched, the resilience may be improved even further.

At the same time, while the general transition to programmable networks may result in several related benefits, future fast-recovery solutions could also be designed in such a way that makes their deployment in legacy and mixed network environments possible and effective, both in terms of the reliability guarantees and the overall cost. Otherwise, the related support in commercial off-the-shelf network equipment might be limited to none, which was among the main shortcomings of many designs proposed to date.

ACKNOWLEDGMENT

The authors would like to thank Thomas Holterbach, James Kempf, Michael Menth, Daniel Merling, Ankit Singla, and Laurent Vanbever for their insightful comments received on the paper draft.

REFERENCES

[1] R. Chirgwin. (2017). *Google Routing Blunder Sent Japan's Internet Dark on Friday.* [Online]. Available: https://www.theregister.co.uk/2017/08/27/google_routing_blunder_sent_japans_internet_dark/

[2] D. Tweney. (2013). *5-Minute Outage Costs Google $545,000 in Revenue*. [Online]. Available: http://venturebeat.com/2013/08/16/3-minute-outage-costs-google-545000-in-revenue/

[3] G. Corfield. (2018). *British Airways' Latest Total Inability to Support Upwardness of Planes Caused by Amadeus System Outage*. [Online]. Available: https://www.theregister.co.uk/2018/07/19/amadeus_british_airways_outage_load_sheet/

[4] C. Gibbs. (2017). *ATT's 911 Outage Result of Mistakes Made by ATT, FCC's Pai Says*. [Online]. Available: https://www.fiercewireless.com/wireless/at-t-s-911-outage-result-mistakes-made-by-at-t-fcc-s-pai-says

[5] J. Young and T. Barth, *Web Performance Analytics Show Even 100-Millisecond Delays Can Impact Customer Engagement and Online Revenue*, Akamai, Cambridge, MA, USA, 2017.

[6] J. Saldan, *Delay Limits for Real-Time Services*, IETF, Fremont, CA, USA, 2016.

[7] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 350–361, 2011.

[8] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, pp. 515–528, Oct. 1998.

[9] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, "An approach to alleviate link overload as observed on an IP backbone," in *Proc. IEEE INFOCOM*, vol. 1, 2003, pp. 406–416.

[10] J. Moy, "OSPF version 2," IETF, Fremont, CA, USA, RFC 2328, Apr. 1998. [Online]. Available: https://tools.ietf.org/html/rfc2328

[11] *Intermediate System-to-Intermediate System (IS-IS) Routing Protocol*, ISO/IEC Standard 10589, 2002.

[12] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 63–74, 2011.

[13] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 115–126, 2012.

[14] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Architect. Protocols Comput. Commun.*, 2012, pp. 139–150.

[15] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving subsecond IGP convergence in large IP networks," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 35–44, Jul. 2005. [Online]. Available: https://doi.org/10.1145/1070873.1070877

[16] J. Liu, A. Panda, A. Singla, B. Godfrey, M. Schapira, and S. Shenker, "Ensuring connectivity via data plane mechanisms," in presented at the 10th USENIX Symp. Netw. Syst. Design Implement. (NSDI), Lombard, IL, USA, 2013, pp. 113–126. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/liu_junda

[17] A. Greenberg *et al.*, "A clean slate 4D approach to network control and management," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 41–54, Oct. 2005. [Online]. Available: http://doi.acm.org/10.1145/1096536.1096541

[18] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[19] H. Yan, D. A. Maltz, T. E. Ng, H. Gogineni, H. Zhang, and Z. Cai, "Tesseract: A 4D network control plane," in *Proc. NSDI*, vol. 7, 2007, pp. 27–27.

[20] J. Feigenbaum, B. Godfrey, A. Panda, M. Schapira, S. Shenker, and A. Singla, "Brief announcement: On the resilience of routing tables," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2012, pp. 237–238.

[21] D. Stamatelakis and W. D. Grover, "IP layer restoration and network planning based on virtual protection cycles," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1938–1949, Oct. 2000.

[22] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "FlowBender: Flow-level adaptive routing for improved latency and throughput in datacenter networks," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2014, pp. 149–160. [Online]. Available: http://doi.acm.org/10.1145/2674005.2674985

[23] J. Papan, P. Segec, M. Moravcik, M. Kontsek, L. Mikus, and J. Uramova, "Overview of IP fast reroute solutions," in *Proc. 16th Int. Conf. Emerg. e-Learn. Technol. Appl. (ICETA)*, Nov. 2018, pp. 417–424.

[24] A. Jarry, "Fast reroute paths algorithms," *Telecommun. Syst.*, vol. 52, no. 2, pp. 881–888, 2013.

[25] A. Kamisiński, "Evolution of IP fast-reroute strategies," in *Proc. 10th Int. Workshop Resilient Netw. Design Model. (RNDM)*, Aug. 2018, pp. 1–6.

[26] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to RSVP-TE for LSP tunnels," IETF, Fremont, CA, USA, RFC 4090, May 2005. [Online]. Available: https://tools.ietf.org/html/rfc4090

[27] Switch Specification 1.3.1. (2013). *OpenFlow*. [Online]. Available: https://bit.ly/2VjOO77

[28] Cisco. (Oct. 2017). *Configuring BGP PIC Edge and Core for IP and MPLS*. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bgp/configuration/xe-3s/irg-xe-3s-book/irg-bgp-mp-pic.html

[29] D. Xu, Y. Xiong, C. Qiao, and G. Li, "Failure protection in layered networks with shared risk link groups," *IEEE Netw.*, vol. 18, no. 3, pp. 36–41, May/Jun. 2004.

[30] P. Sebos, J. Yates, G. Hjalmtysson, and A. Greenberg, "Auto-discovery of shared risk link groups," in *Proc. Opt. Fiber Commun. Conf. Exhibit. (OFC)*, vol. 3, 2001, pp. 1–3.

[31] M. Menth, M. Duelli, R. Martin, and J. Milbrandt, "Resilience analysis of packet-switched communication networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1950–1963, Dec. 2009.

[32] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: Loop-free alternates," IETF, Fremont, CA, USA, RFC 5286, Sep. 2008. [Online]. Available: https://tools.ietf.org/html/rfc5286

[33] T. Elhourani, A. Gopalan, and S. Ramasubramanian, "IP fast rerouting for multi-link failures," in *Proc. IEEE INFOCOM*, 2014, pp. 2148–2156.

[34] M. Golash, "Reliability in Ethernet networks: A survey of various approaches," *Bell Labs Techn. J.*, vol. 11, no. 3, pp. 161–171, 2006.

[35] M. Gjoka, V. Ram, and X. Yang, "Evaluation of IP fast reroute proposals," in *Proc. 2nd Int. Conf. Commun. Syst. Softw. Middleware*, Jan. 2007, pp. 1–8.

[36] J. Papán, P. Segeć, and P. Palüch, "Analysis of existing IP fast reroute mechanisms," in *Proc. Int. Conf. Inf. Digit. Technol.*, Jul. 2015, pp. 291–297.

[37] A. Raj and O. C. Ibe, "A survey of IP and multiprotocol label switching fast reroute schemes," *Comput. Netw.*, vol. 51, no. 8, pp. 1882–1907, 2007.

[38] L. Jorge and T. Gomes, "Survey of recovery schemes in MPLS networks," in *Proc. IEEE Int. Conf. Depend. Comput. Syst.*, 2006, pp. 110–118.

[39] R. B. da Silva and E. S. Mota, "A survey on approaches to reduce BGP interdomain routing convergence delay on the Internet," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2949–2984, 4th Quart., 2017.

[40] A. S. da Silva, P. Smith, A. Mauthe, and A. Schaeffer-Filho, "Resilience support in software-defined networking: A survey," *Comput. Netw.*, vol. 92, pp. 189–207, Dec. 2015.

[41] M. F. Habib, M. Tornatore, F. Dikbiyik, and B. Mukherjee, "Disaster survivability in optical communication networks," *Comput. Commun.*, vol. 36, no. 6, pp. 630–644, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366413000224

[42] P. Cholda and A. Jajszczyk, "Recovery and its quality in multilayer networks," *J. Lightw. Technol.*, vol. 28, no. 4, pp. 372–389, Feb. 15, 2010.

[43] S. D. Maesschalck *et al.*, "PAN–European optical transport networks: An availability-based comparison," *Photon. Netw. Commun.*, vol. 5, no. 3, pp. 203–225, 2003.

[44] J. Rak *et al.*, "RECODIS: Resilient communication services protecting end-user applications from disaster-based failures," in *Proc. 18th Int. Conf. Transp. Opt. Netw. (ICTON)*, 2016, pp. 1–4.

[45] J. Rak and D. Hutchison, *Guide to Disaster-Resilient Communication Networks*. Cham, Switzerland: Springer, 2020.

[46] Y. Rekhter, S. Hares, and T. Li, "A border gateway protocol 4 (BGP-4)," IETF, Fremont, CA, USA, RFC 4271, Jan. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4271

[47] S. Rai, B. Mukherjee, and O. Deshpande, "IP resilience within an autonomous system: Current approaches, challenges, and future directions," *IEEE Commun. Mag.*, vol. 43, no. 10, pp. 142–149, Oct. 2005.

[48] R. Stankiewicz, P. Cholda, and A. Jajszczyk, "QoX: What is it really?" *IEEE Commun. Mag.*, vol. 49, no. 4, pp. 148–158, Apr. 2011.

[49] "Quality of service mapping and interconnection between Ethernet, Internet protocol and multiprotocol label switching networks," Int. Telecommun. Union, Geneva, Switzerland, ITU-T Recommendation Y.1566, 2012.

[50] W. C. Hardy, *"QoS" Measurement and Evaluation of Telecommunications Quality of Service*. Hoboken, NJ, USA: Wiley, 2001.

[51] J. Gozdecki, A. Jajszczyk, and R. Stankiewicz, "Quality of service terminology in IP networks," *IEEE Commun. Mag.*, vol. 41, no. 3, pp. 153–159, Mar. 2003.

[52] "Internet protocol data communication service—IP packet transfer and availability performance parameters," Int. Telecommun. Union, Geneva, Switzerland, ITU-T Recommendation Y.1540, 2019.

[53] "Network performance objectives for IP-based services," Int. Telecommun. Union, Geneva, Switzerland, ITU-T Recommendation Y.1541, 2011.

[54] A. F. Hansen, T. Čičic, and S. Gjessing, "Alternative schemes for proactive IP recovery," in *Proc. 2nd Conf. Next Gener. Internet Design Eng. (NGI)*, 2006, pp. 1–8.

[55] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. F. Hansen, "Fast recovery from dual-link or single-node failures in IP networks using tunneling," *IEEE/ACM Trans. Netw.*, vol. 18, no. 6, pp. 1988–1999, Dec. 2010.

[56] J. P. G. Sterbenz *et al.*, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Comput. Netw.*, vol. 54, no. 8, pp. 1245–1265, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610000824

[57] J. P. G. Sterbenz, E. K. Cetinkaya, M. A. Hameed, A. Jabbar, S. Qian, and J. P. Rohrer, "Evaluation of network resilience, survivability, and disruption tolerance: Analysis, topology generation, simulation, and experimentation," *Telecommun. Syst.*, vol. 52, no. 2, pp. 705–736, 2013.

[58] J. Rak, *Resilient Routing in Communication Networks* (Computer Communications and Networks), 1st ed. Cham, Switzerland: Springer, 2015.

[59] A. Mauthe *et al.*, "Disaster-resilient communication networks: Principles and best practices," in *Proc. 8th Int. Workshop Resilient Netw. Design Model. (RNDM)*, Halmstad, Sweden, Sep. 2016, pp. 13–15.

[60] S. Dobson, D. Hutchison, A. Mauthe, A. Schaeffer-Filho, P. Smith, and J. P. G. Sterbenz, "Self-organization and resilience for networked systems: Design principles and open research issues," *Proc. IEEE*, vol. 107, no. 4, pp. 819–834, 2019.

[61] T. Gomes *et al.*, "A survey of strategies for communication networks to protect against large-scale natural disasters," in *Proc. 8th Int. Workshop Resilient Netw. Design Model. (RNDM)*, Halmstad, Sweden, Sep. 2016, pp. 13–15.

[62] A. Pašić *et al.*, "eFRADIR: An enhanced framework for disaster resilience," *IEEE Access*, vol. 9, pp. 13125–13148, 2021.

[63] M. J. Khabbaz, C. M. Assi, and W. F. Fawaz, "Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 607–640, 2nd Quart., 2012.

[64] A. Avizienis, J.-C. Laprie, and B. Randell, "Dependability and its threats: A taxonomy," in *Building the Information Society* (IFIP International Federation for Information Processing), vol. 156, R. Jacquart, Ed. Boston, MA, USA: Springer, 2004, pp. 91–120.

[65] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Depend. Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.

[66] P. Cholda, J. Tapolcai, T. Cinkler, K. Wajda, and A. Jajszczyk, "Quality of resilience as a network reliability characterization tool," *IEEE Netw.*, vol. 23, no. 2, pp. 11–19, Mar. 2009.

[67] A. Autenrieth and A. Kirstadter, "Engineering end-to-end IP resilience using resilience-differentiated QoS," *IEEE Commun. Mag.*, vol. 40, no. 1, pp. 50–57, Jan. 2002.

[68] P. Cholda, A. Mykkeltveit, B. E. Helvik, O. J. Wittner, and A. Jajszczyk, "A survey of resilience differentiation frameworks in communication networks," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 4, pp. 32–55, 3rd Quart., 2007.

[69] C. Huang, V. Sharma, K. Owens, and S. Makam, "Building reliable MPLS networks using a path protection mechanism," *IEEE Commun. Mag.*, vol. 40, no. 3, pp. 156–162, Mar. 2002.

[70] F. A. Hellstrand and V. Sharma, "Framework for multi-protocol label switching (MPLS)-based recovery," IETF, Fremont, CA, USA, RFC 3469, Feb. 2003. [Online]. Available: https://tools.ietf.org/html/rfc3469

[71] A. Autenrieth, "Recovery time analysis of differentiated resilience in MPLS," in *Proc. 4th Int. Workshop Design Rel. Commun. Netw. (DRCN)*, 2003, pp. 333–340.

[72] G. Ellinas, D. Papadimitriou, J. Rak, D. Staessens, J. P. G. Sterbenz, and K. Walkowiak, "Practical issues for the implementation of survivability and recovery techniques in optical networks," *Opt. Switch. Netw.*, vol. 14, pp. 179–193, Aug. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1573427714000368

[73] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. San Francisco, CA, USA: Morgan Kaufmann, 2004.

[74] A. Dusia and A. S. Sethi, "Recent advances in fault localization in computer networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 3030–3051, 4th Quart., 2016.

[75] *G.975: Forward Error Correction for Submarine Systems*, Int. Telecommun. Union, Geneva, Switzerland, 2000.

[76] C. Mas-Machuca and P. Thiran, "An efficient algorithm for locating soft and hard failures in WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1900–1911, Oct. 2000.

[77] S. Q. Zhuang, D. Geels, I. Stoica, and R. H. Katz, "On failure detection algorithms in overlay networks," in *Proc. IEEE 24th Annu. Joint Conf. Comput. Commun. Soc.*, vol. 3, 2005, pp. 2112–2123.

[78] D. Katz and D. Ward, "Bidirectional forwarding detection (BFD)," IETF, Fremont, CA, USA, RFC 5880, Jun. 2010. [Online]. Available: https://tools.ietf.org/html/rfc5880

[79] R. Steinert and D. Gillblad, "Towards distributed and adaptive detection and localisation of network faults," in *Proc. 6th Adv. Int. Conf. Telecommun.*, 2010, pp. 384–389.

[80] R. Cohen and G. Nakibly, "Maximizing restorable throughput in MPLS networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 568–581, Apr. 2010.

[81] *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges*, IEEE Standard 802.1D-2004, pp. 1–281, Jun. 2004. [Online]. Available: https://standards.ieee.org/standard/802_1D-2004.html

[82] J. Qiu, M. Gurusamy, K. C. Chua, and Y. Liu, "Local restoration with multiple spanning trees in metro Ethernet," in *Proc. Int. Conf. Opt. Netw. Design Model.*, 2008, pp. 1–6.

[83] K. Elmeleegy, A. L. Cox, and T. S. E. Ng, "On count-to-infinity induced forwarding loops in Ethernet networks," in *Proc. IEEE INFOCOM 25th IEEE Int. Conf. Comput. Commun.*, Apr. 2006, pp. 1–13.

[84] L. Su, W. Chen, H. Su, Z. Xiao, D. Jin, and L. Zeng, "Ethernet ultra fast switching: A tree-based local recovery scheme," in *Proc. 11th IEEE Singapore Int. Conf. Commun. Syst.*, 2008, pp. 314–318.

[85] *Part 3: Media Access Control (MAC) Bridges: Amendment 2–Rapid Reconfiguration*, IEEE Standard 802.1w-2001, pp. 1–116, Jul. 2001, [Online]. Available: https://standards.ieee.org/standard/802_1w-2001.html

[86] *IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks*, IEEE Standardb 802.1Q-2003, May 2003, pp. 1–322.

[87] A. Gopalan and S. Ramasubramanian, "Fast recovery from link failures in Ethernet networks," in *Proc. 9th Int. Conf. Design Rel. Commun. Netw. (DRCN)*, 2013, pp. 1–10.

[88] A. Gopalan and S. Ramasubramanian, "Fast recovery from link failures in Ethernet networks," *IEEE Trans. Rel.*, vol. 63, no. 2, pp. 412–426, Jun. 2014.

[89] *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges*, IEEE Standard 802.1D-1990, Mar. 1991, pp. 1–176.

[90] *IEEE Standard for Local Area Network MAC (Media Access Control) Bridges*, ANSI/IEEE Standard 802.1D, Dec. 1998, pp. 1–373.

[91] *IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks*, IEEE Standard 802.1Q-1998, Mar. 1999, pp. 1–214.

[92] S. Varadarajan and T. Chiueh, "Automatic fault detection and recovery in real time switched Ethernet networks," in *Proc. IEEE Conf. Comput. Commun. 18th Annu. Joint Conf. IEEE Comput. Commun. Soc. Future Now (INFOCOM)*, vol. 1, 1999, pp. 161–169.

[93] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh, "Viking: A multi-spanning-tree Ethernet architecture for metropolitan area and cluster networks," in *Proc. IEEE INFOCOM*, vol. 4, 2004, pp. 2283–2294.

[94] R. Pallos, J. Farkas, I. Moldovan, and C. Lukovszki, "Performance of rapid spanning tree protocol in access and metro networks," in *Proc. 2nd Int. Conf. Access Netw. Workshops*, 2007, pp. 1–8.

[95] D. Jin, W. Chen, Z. Xiao, and L. Zeng, "Single link switching mechanism for fast recovery in tree-based recovery schemes," in *Proc. Int. Conf. Telecommun.*, 2008, pp. 1–5.

[96] D. Jin, Y. Li, W. Chen, L. Su, and L. Zeng, "Ethernet ultra-fast switching: A tree-based local recovery scheme," *IET Commun.*, vol. 4, no. 4, pp. 410–418, 2010.

[97] J. Qiu, M. Gurusamy, K. C. Chua, and Y. Liu, "Local restoration with multiple spanning trees in metro Ethernet networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 2, pp. 602–614, Apr. 2011.

[98] J. Qiu, Y. Liu, G. Mohan, and K. C. Chua, "Fast spanning tree reconnection for resilient metro Ethernet networks," in *Proc. IEEE Int. Conf. Commun.*, 2009, pp. 1–5.

[99] M. Terasawa, M. Nishida, S. Shimizu, Y. Arakawa, S. Okamoto, and N. Yamanaka, "Recover-forwarding method in link failure with pre-established recovery table for wide area Ethernet," in *Proc. IEEE Int. Conf. Commun.*, 2009, pp. 1–5.

[100] J. Qiu, G. Mohan, K. C. Chua, and Y. Liu, "Handling double-link failures in metro Ethernet networks using fast spanning tree reconnection," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, 2009, pp. 1–6.

[101] J. Farkas and Z. Arato, "Performance analysis of shortest path bridging control protocols," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, 2009, pp. 1–6.

[102] *IEEE Standard for Local and Metropolitan Area Networks—Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks*, IEEE Standard 802.1Q-2011, pp. 1–1365, Aug. 2011.

[103] D. M. Shan, C. K. Chiang, G. Mohan, and J. Qiu, "Partial spatial protection for differentiated reliability in FSTR-based metro Ethernet networks," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, 2011, pp. 1–5.

[104] J. McCauley, A. Sheng, E. J. Jackson, B. Raghavan, S. Ratnasamy, and S. Shenker, "Taking an AXE to L2 spanning trees," in *Proc. 14th ACM Workshop Hot Topics Netw. (HotNets-XIV)*, 2015, p. 15. [Online]. Available: https://doi.org/10.1145/2834050.2834097

[105] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 24: Path Control and Reservation*, IEEE Standard 802.1Qca-2015, pp. 1–120, Mar. 2016.

[106] M. Santos and J. Gregoire, "Improving carrier Ethernet recovery time using a fast reroute mechanism," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, 2016, pp. 1–7.

[107] J. McCauley, M. Zhao, E. J. Jackson, B. Raghavan, S. Ratnasamy, and S. Shenker, "The deforestation of L2," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, 2016, pp. 497–510. [Online]. Available: https://doi.org/10.1145/2934872.2934877

[108] *IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks*, IEEE Standard 802.1Q-2005, pp. 1–300, May 2006.

[109] *IEEE Standard for Local and Metropolitan Area Networks—Media Access Control (MAC) Bridges and Virtual Bridges [Edition]*, IEEE Standard 802.1Q-2012, pp. 1–1782, Dec. 2012.

[110] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks*, IEEE Standard 802.1Q-2014, pp. 1–1832, Dec. 2014.

[111] *IEEE Standard for Local and Metropolitan Area Network—Bridges and Bridged Networks*, IEEE Standard 802.1Q-2018, pp. 1–1993, Jul. 2018.

[112] W. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network restoration," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 1998, pp. 537–543.

[113] O. Lemeshko and K. Arous, "Fast ReRoute model for different backup schemes in MPLS-network," in *Proc. 1st Int. Sci. Pract. Conf. Probl. Infocommun. Sci. Technol.*, 2014, pp. 39–41.

[114] J. McManus, J. Malcolm, M. D. O'Dell, D. O. Awduche, and J. Agogbua, "Requirements for traffic engineering over MPLS," IETF, Fremont, CA, USA, RFC 2702, Sep. 1999. [Online]. Available: https://tools.ietf.org/html/rfc2702

[115] D. L. Haskin and R. Krishnan, "A method for setting an alternative label switched paths to handle fast reroute," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-haskin-mpls-fast-reroute-05, Nov. 2000. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-haskin-mpls-fast-reroute-05

[116] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," IETF, Fremont, CA, USA, RFC 3031, Jan. 2001. [Online]. Available: https://tools.ietf.org/html/rfc3031

[117] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," IETF, Fremont, CA, USA, RFC 3209, Dec. 2001. [Online]. Available: https://tools.ietf.org/html/rfc3209

[118] M. Kodialam and T. V. Lakshman, "Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information," in *Proc. IEEE Conf. Comput. Commun. 20th Annu.*

Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM), vol. 1, 2001, pp. 376–385.

[119] L. Hundessa and J. D. Pascual, "Fast rerouting mechanism for a protected label switched path," in *Proc. 10th Int. Conf. Comput. Commun. Netw.*, 2001, pp. 527–530.

[120] E. Mannie and D. Papadimitriou, "Recovery (protection and restoration) terminology for generalized multi-protocol label switching (GMPLS)," IETF, Fremont, CA, USA, RFC 4427, Mar. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4427

[121] D. Papadimitriou and E. Mannie, "Analysis of generalized multi-protocol label switching (GMPLS)-based recovery mechanisms (including protection and restoration)," IETF, Fremont, CA, USA, RFC 4428, Mar. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4428

[122] L. Andersson, I. Minei, and B. Thomas, "LDP specification," IETF, Fremont, CA, USA, RFC 5036, Oct. 2007. [Online]. Available: https://tools.ietf.org/html/rfc5036

[123] J. V. A. Farrel and A. Ayyangar, "Inter-domain MPLS and GMPLS traffic engineering—Resource reservation protocol-traffic engineering (RSVP-TE) extensions," IETF, Fremont, CA, USA, RFC 5151, Feb. 2008. [Online]. Available: https://tools.ietf.org/html/rfc5151

[124] D. Wang and G. Li, "Efficient distributed bandwidth management for MPLS fast reroute," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 486–495, Apr. 2008.

[125] Y. Wang *et al.*, "R3: Resilient routing reconfiguration," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 291–302, Aug. 2010. [Online]. Available: https://doi.org/10.1145/1851275.1851218

[126] A. Hassan, M. Bazama, T. Saad, and H. T. Mouftah, "Investigation of fast reroute mechanisms in an optical testbed environment," in *Proc. 7th Int. Symp. High Capacity Opt. Netw. Enabl. Technol.*, 2010, pp. 247–251.

[127] N. Sprecher and A. Farrel, "MPLS transport profile (MPLSTP) survivability framework," IETF, Fremont, CA, USA, RFC 6372, Sep. 2011. [Online]. Available: https://tools.ietf.org/html/rfc6372

[128] G. Ramachandran, L. Ciavattone, and A. Morton, "Restoration measurements on an IP/MPLS backbone: The effect of Fast Reroute on link failure," in *Proc. IEEE 19th IEEE Int. Workshop Qual. Service*, 2011, pp. 1–6.

[129] K. Koushik, R. Cetin, and T. Nadeau, "Multiprotocol label switching (MPLS) traffic engineering management information base for fast reroute," IETF, Fremont, CA, USA, RFC 6445, Nov. 2011. [Online]. Available: https://tools.ietf.org/html/rfc6445

[130] S. Bryant, S. Previdi, and M. Shand, "A framework for IP and MPLS fast reroute using Not-Via addresses," IETF, Fremont, CA, USA, RFC 6981, Aug. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6981

[131] T. Benhcine, H. Elbiaze, and K. Idoudi, "Fast reroute-based network resiliency experimental investigations," in *Proc. 15th Int. Conf. Transp. Opt. Netw. (ICTON)*, 2013, pp. 1–4.

[132] O. Lemeshko, A. Romanyuk, and H. Kozlova, "Design schemes for MPLS fast reroute," in *Proc. 12th Int. Conf. Exp. Design. Appl. CAD Syst. Microelectron. (CADSM)*, 2013, pp. 202–203.

[133] M. Taillon, T. Saad, R. Gandhi, Z. Ali, and M. Bhatia, "Updates to the resource reservation protocol for fast reroute of traffic engineering GMPLS Label Switched Paths (LSPs)," IETF, Fremont, CA, USA, RFC 8271, Oct. 2017. [Online]. Available: https://tools.ietf.org/html/rfc8271

[134] O. S. Yeremenko, O. V. Lemeshko, and N. Tariki, "Fast ReRoute scalable solution with protection schemes of network elements," in *Proc. IEEE 1st Ukraine Conf. Elect. Comput. Eng. (UKRCON)*, 2017, pp. 783–788.

[135] S. Schmid and J. Srba, "Polynomial-time what-if analysis for prefix-manipulating MPLS networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 1799–1807.

[136] J. S. Jensen, T. B. Krøgh, J. S. Madsen, S. Schmid, J. Srba, and M. T. Thorgersen, "P-Rex: Fast verification of MPLS networks with multiple link failures," in *Proc. 14th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2018, pp. 217–227. [Online]. Available: https://doi.org/10.1145/3281411.3281432

[137] O. Lemeshko and O. Yeremenko, "Linear optimization model of MPLS traffic engineering Fast ReRoute for link, node, and bandwidth protection," in *Proc. 14th Int. Conf. Adv. Trends Radioelecrtron. Telecommun. Comput. Eng. (TCSET)*, 2018, pp. 1009–1013.

[138] Y. Shen, J. M. Jeganathan, B. Decraene, H. Gredler, C. Michel, and H. Chen, "MPLS egress protection framework," IETF, Fremont, CA, USA, RFC 8679, Dec. 2019. [Online]. Available: https://tools.ietf.org/html/rfc8679

[139] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 749–762, Aug. 2008.

[140] J. S. Arora, *Introduction to Optimum Design*, 4th ed. Boston, MA, USA: Academic, 2017.

[141] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs," in *Proc. Conf. Appl. Technol. Architect. Protocols Comput. Commun. (SIGCOMM)*, 2003, pp. 313–324. [Online]. Available: http://doi.acm.org/10.1145/863955.863991

[142] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: Traffic engineering in dynamic networks," in *Proc. Conf. Appl. Technol. Architect. Protocols Comput. Commun. (SIGCOMM)*, 2006, pp. 99–110. [Online]. Available: http://doi.acm.org/10.1145/1159913.1159926

[143] P. G. Jensen, M. Konggaard, D. Kristiansen, S. Schmid, B. C. Schrenk, and J. Srba, "AalWiNes: A fast and quantitative what-if analysis tool for MPLS networks," in *Proc. 16th ACM Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2020, pp. 474–481.

[144] C. J. Anderson *et al.*, "NetKAT: Semantic foundations for networks," *ACM SIGPLAN Notices*, vol. 49, no. 1, pp. 113–126, 2014.

[145] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks," in presented at the 9th USENIX Symp. Netw. Syst. Design Implement. (NSDI), 2012, pp. 113–126.

[146] L. Hundessa and J. Domingo-Pascual, "Reliable and fast rerouting mechanism for a protected label switched path," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 2, Nov. 2002, pp. 1608–1612.

[147] M. Menth, A. Reifert, and J. Milbrandt, "Self-protecting multipaths—A simple and resource-efficient protection switching mechanism for MPLS networks," in *Proc. Netw. Lecture Notes in Computer Science*, vol. 3042, 2004, pp. 526–537.

[148] M. Menth, R. Martin, and U. Sporlein, "Failure-specific self-protecting multipaths—Increased capacity savings or overengineering?" in *Proc. 6th Int. Workshop Design Rel. Commun. Netw.*, 2007, pp. 1–7.

[149] M. Menth, R. Martin, and U. Sporlein, "Optimization of the self-protecting multipath for deployment in legacy networks," in *Proc. IEEE Int. Conf. Commun.*, 2007, pp. 421–427.

[150] M. Shand and S. Bryant, "IP fast reroute framework," IETF, Fremont, CA, USA, RFC 5714, Jan. 2010. [Online]. Available: https://tools.ietf.org/html/rfc5714

[151] R. Callon, "TCP and UDP with bigger addresses (TUBA), a simple proposal for Internet addressing and routing," IETF, Fremont, CA, USA, RFC 1347, Jun. 1992. [Online]. Available: https://tools.ietf.org/html/rfc1347

[152] K. W. Kwong, L. Gao, R. Guerin, and Z. L. Zhang, "On the feasibility and efficacy of protection routing in IP networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[153] K. W. Kwong, L. Gao, R. Guerin, and Z. L. Zhang, "On the feasibility and efficacy of protection routing in IP networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 5, pp. 1543–1556, Oct. 2011.

[154] G. Enyedi, A. Csaszar, A. Atlas, C. Bowers, and A. Gopalan, "An algorithm for computing IP/LDP fast reroute using maximally redundant trees (MRT-FRR)," IETF, Fremont, CA, USA, RFC 7811, Jun. 2016. [Online]. Available: http://tools.ietf.org/rfc/rfc7811

[155] A. Atlas, C. Bowers, and G. Enyedi, "An architecture for IP/LDP fast reroute using maximally redundant trees (MRT-FRR)," IETF, Fremont, CA, USA, RFC 7812, Jun. 2016. [Online]. Available: https://tools.ietf.org/html/rfc7812

[156] T. Čičic, A. F. Hansen, and O. K. Apeland, "Redundant trees for fast IP recovery," in *Proc. Broadnets*, 2007, pp. 152–159.

[157] C. Alaettinoglu and V. Jacobson, "Towards milli-second IGP convergence," IETF, Fremont, CA, USA, Internet-Draft draft-alaettinoglu-isis-convergence-00, Nov. 2000. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-alaettinoglu-isis-convergence-00

[158] A. Shaikh, C. Isett, A. Greenberg, M. Roughan, and J. Gottlieb, "A case study of OSPF behavior in a large enterprise network," in *Proc. ACM IMW*, 2002, pp. 217–230.

[159] J. Papan, P. Segec, and P. Paluch, "Utilization of PIM-DM in IP fast reroute," in *Proc. IEEE 12th Int. Conf. Emerg. e-Learn. Technol. Appl. (ICETA)*, Dec. 2014, pp. 373–378.

[160] G. Schollmeier *et al.*, "Improving the resilience in IP networks," in *Proc. Workshop High Perform. Switch. Routing (HPSR)*, Jun. 2003, pp. 91–96.

[161] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets," in *Proc. Conf. Appl. Technol. Architect. Protocols Comput. Commun. (SIGCOMM)*, 2007, pp. 241–252. [Online]. Available: http://doi.acm.org/10.1145/1282380.1282408

[162] B. Yang, J. Liu, S. Shenker, J. Li, and K. Zheng, "Keep Forwarding: Towards *k*-link failure resilient routing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2014, pp. 1617–1625.

[163] M. Chiesa *et al.*, "On the resiliency of static forwarding tables," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1133–1146, Apr. 2017. [Online]. Available: https://doi.org/10.1109/TNET.2016.2619398

[164] M. Chiesa *et al.*, "On the resiliency of randomized routing against multiple edge failures," in *Proc. 43rd Int. Colloquium Automata Lang. Program. (ICALP)*, vol. 55, 2016, p. 134. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2016/6269

[165] M. Chiesa *et al.*, "The quest for resilient (static) forwarding tables," in *Proc. IEEE 35th Annu. Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[166] L. Csikor and G. Rétvári, "IP fast reroute with remote loop-free alternates: The unit link cost case," in *Proc. IV Int. Congr. Ultra Mod. Telecommun. Control Syst.*, Oct. 2012, pp. 663–669.

[167] L. Csikor and G. Rétvári, "On providing fast protection with remote loop-free alternates," *Telecommun. Syst.*, vol. 60, no. 4, pp. 485–502, Dec. 2015. [Online]. Available: https://doi.org/10.1007/s11235-015-0006-9

[168] P. Francois and O. Bonaventure, "An evaluation of IP-based fast reroute techniques," in *Proc. ACM Int. Conf. Emerg. Netw. Exp. Technol.*, 2005, pp. 244–245.

[169] M. Shand and S. Bryant, "A framework for loop-free convergence," IETF, Fremont, CA, USA, RFC 5715, Jan. 2010. [Online]. Available: https://tools.ietf.org/html/rfc5715

[170] R. Teixeira and J. Rexford, "Managing routing disruptions in Internet service provider networks," *Comm. Mag.*, vol. 44, no. 3, pp. 160–165, Mar. 2006. doi: 10.1109/MCOM.2006.1607880.

[171] F. Clad, P. Merindol, J.-J. Pansiot, P. Francois, and O. Bonaventure, "Graceful convergence in link-state IP networks: A lightweight algorithm ensuring minimal operational impact," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 300–312, Feb. 2014.

[172] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *Proc. IEEE INFOCOM*, vol. 4, 2004, pp. 2307–2317.

[173] S. Nelakuditi, S. Lee, Y. Yu, and Z.-L. Zhang, "Failure insensitive routing for ensuring service availability," in *Quality of Service—IWQoS 2003*, K. Jeffay, I. Stoica, and K. Wehrle, Eds. Berlin, Germany: Springer, 2003, pp. 287–304.

[174] A. Kvalbein, A. F. Hansen, T. Čičic, S. Gjessing, and O. Lysne, "Fast recovery from link failures using resilient routing layers," in *Proc. 10th IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2005, pp. 554–560.

[175] A. Kvalbein, A. F. Hansen, T. Čičic, S. Gjessing, and O. Lysne, "Fast IP network recovery using multiple routing configurations," in *Proc. IEEE INFOCOM 25th Int. Conf. Comput. Commun.*, Apr. 2006, pp. 1–11.

[176] J. Wang and S. Nelakuditi, "IP fast reroute with failure inferencing," in *Proc. SIGCOMM Workshop Internet Netw. Manag. (INM)*, 2007, pp. 268–273. [Online]. Available: http://doi.acm.org/10.1145/1321753.1321764

[177] J. Tapolcai and G. Rétvári, "Router virtualization for improving IP-level resilience," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 935–943.

[178] S. Bryant, C. Filsfils, S. Previdi, M. Shand, and N. So, "Remote loop-free alternate (LFA) fast reroute (FRR)," RFC 7490, Apr. 2015. [Online]. Available: https://tools.ietf.org/html/rfc7490

[179] J. Tapolcai, G. Rétvári, P. Babarczi, and E. R. Bérczi-Kovács, "Scalable and efficient multipath routing via redundant trees," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 982–996, May 2019.

[180] C. Filsfils *et al.*, "Loop-free alternate (LFA) applicability in service provider (SP) networks," IETF, Fremont, CA, USA, RFC 6571, Jun. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6571

[181] *Cisco IOS XR Routing Configuration Guide for the Cisco CRS Router, Release 4.2*, Cisco Syst., San Jose, CA, USA, 2012.

[182] Hewlett-Packard. (2008). *HP 6600 Router Series: QuickSpecs*. [Online]. Available: http://h18000.www1.hp.com/products/quickspecs/13811_na/13811_na.PDF

[183] *JUNOS 12.3 Routing Protocols Configuration Guide*, Juniper Netw., Sunnyvale, CA, USA, 2012.

[184] G. Rétvári, J. Tapolcai, G. Enyedi, and A. Császár, "IP fast ReRoute: Loop free alternates revisited," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 2948–2956.

[185] D. Hock, M. Hartmann, C. Schwartz, and M. Menth, "Effectiveness of link cost optimization for IP rerouting and IP fast reroute," in *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, B. Müller-Clostermann, K. Echtle, and E. P. Rathgeb, Eds. Heidelberg, Germany: Springer, 2010, pp. 78–90.

[186] L. Csikor, M. Nagy, and G. Rétvári, "Network optimization techniques for improving fast IP-level resilience with loop-free alternates," *Infocommun. J.*, vol. 3, no. 4, pp. 2–10, Dec. 2011. [Online]. Available: http://eprints.gla.ac.uk/131074/

[187] L. Csikor, J. Tapolcai, and G. Rétvári, "Optimizing IGP link costs for improving IP-level resilience with loop-free alternates," *Comput. Commun.*, vol. 36, no. 6, pp. 645–655, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366412003167

[188] M. Nagy, J. Tapolcai, and G. Rétvári, "Optimization methods for improving IP-level fast protection for local shared risk groups with Loop-Free Alternates," *Telecommun. Syst.*, vol. 56, no. 1, pp. 103–119, May 2014. [Online]. Available: https://doi.org/10.1007/s11235-013-9822-y

[189] M. Hartmann, D. Hock, and M. Menth, "Routing optimization for IP networks with loop-free alternates," *Comput. Netw.*, vol. 95, pp. 35–50, Feb. 2016.

[190] S. Litkowski, B. Decraene, C. Filsfils, and P. Francois, "Micro-loop prevention by introducing a local convergence delay," IETF, Fremont, CA, USA, RFC 8333, Mar. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8333

[191] G. Enyedi, G. Rétvári, and T. Cinkler, "A novel loop-free IP fast reroute algorithm," in *Dependable and Adaptable Networks and Services*, A. Pras and M. van Sinderen, Eds. Heidelberg, Germany: Springer, 2007, pp. 111–119.

[192] W. Braun and M. Menth, "Loop-free alternates with loop detection for fast reroute in software-defined carrier and data center networks," *J. Netw. Syst. Manag.*, vol. 24, no. 3, pp. 470–490, Jul. 2016. [Online]. Available: https://doi.org/10.1007/s10922-016-9369-9

[193] A. Atlas, "U-turn alternates for IP/LDP fast-reroute," IETF, Fremont, CA, USA, Internet-Draft, Feb. 2006. [Online]. Available: https://tools.ietf.org/pdf/draft-atlas-ip-local-protect-uturn-03.pdf

[194] F. Baker and P. Savola, "Ingress filtering for multihomed networks," IETF, Fremont, CA, USA, RFC 3704, Mar. 2004. [Online]. Available: https://tools.ietf.org/html/rfc3704

[195] B. Zhang, J. Wu, and J. Bi, "RPFP: IP fast ReRoute with providing complete protection and without using tunnels," in *Proc. IEEE/ACM 21st Int. Symp. Qual. Service (IWQoS)*, Jun. 2013, pp. 1–10.

[196] P. Francois, "Improving the convergence of IP routing protocols," Ph.D. dissertation, Département d'Ingénierie Informatique, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 2018. [Online]. Available: http://biblio.info.ucl.ac.be/2007/457147.pdf

[197] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment routing architecture," IETF, Fremont, CA, USA, RFC 8402, Jul. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8402

[198] S. Bryant, C. Filsfils, S. Previdi, and M. Shand, "IP fast reroute using tunnels," IETF, IETF, Fremont, CA, USA, Internet-Draft, Nov. 2007. [Online]. Available: https://tools.ietf.org/pdf/draft-bryant-ipfrr-tunnels-03.pdf

[199] S. Litkowski *et al.*, "Topology independent fast reroute using segment routing," Internet Eng. Task Force, IETF, Fremont, CA, USA, Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa-03, Mar. 2020. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rtgwg-segment-routing-ti-lfa-03

[200] G. Enyedi, G. Rétvári, P. Szilágyi, and A. Császár, "IP fast ReRoute: Lightweight not-via," in *Networking*, L. Fratta, H. Schulzrinne, Y. Takahashi, and O. Spaniol, Eds. Berlin, Germany: Springer, 2009, pp. 157–168.

[201] G. Enyedi, P. Szilágyi, G. Rétvári, and A. Császár, "IP fast ReRoute: Lightweight not-via without additional addresses," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 2771–2775.

[202] G. Enyedi, "Novel algorithms for IP fast reroute," Ph.D. dissertation, Dept. Telecommun. Media Informat., Budapest Univ. Technol. Econ., Budapest, Hungary, 2011.

[203] M. Menth, M. Hartmann, R. Martin, T. Cicic, and A. Kvalbein, "Loop-free alternates and not-via addresses: A proper combination for IP fast reroute?" *Comput. Netw.*, vol. 54, no. 8, pp. 1300–1315, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128609003491

[204] J. Papán, P. Segeč, and P. Palúch, "Tunnels in IP fast reroute," in *Proc. 10th Int. Conf. Digit. Technol.*, Jul. 2014, pp. 270–274.

[205] M. Xu, Q. Li, L. Pan, Q. Li, and D. Wang, "Minimum protection cost tree: A tunnel-based IP fast ReRoute scheme," *Comput. Commun.*, vol. 35, no. 17, pp. 2082–2092, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366412002137

[206] A. Li, P. Francois, and X. Yang, "On improving the efficiency and manageability of NotVia," in *Proc. ACM CoNEXT Conf. (CoNEXT)*, 2007, pp. 1–26. [Online]. Available: http://doi.acm.org/10.1145/1364654.1364688

[207] M. Medard, S. G. Finn, R. A. Barry, and R. G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Trans. Netw.*, vol. 7, no. 5, pp. 641–652, Oct. 1999.

[208] K. Xi and H. Chao, "ESCAP: Efficient scan for alternate paths to achieve IP fast rerouting," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Nov. 2007, pp. 1860–1865.

[209] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah, "Fast local rerouting for handling transient link failures," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 359–372, Apr. 2007.

[210] G. Enyedi and G. Rétvári, "A loop-free interface-based fast reroute technique," in *Proc. Next Gener. Internet Netw.*, Apr. 2008, pp. 39–44.

[211] S. Antonakopoulos, Y. Bejerano, and P. Koppol, "A simple IP fast reroute scheme for full coverage," in *Proc. IEEE 13th Int. Conf. High Perform. Switch. Routing*, Jun. 2012, pp. 15–22.

[212] K.-T. Foerster, Y.-A. Pignolet, S. Schmid, and G. Tredan, "Local fast failover routing with low stretch," *SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 1, pp. 35–41, Apr. 2018. [Online]. Available: https://doi.org/10.1145/3211852.3211858

[213] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, "Failure inferencing based fast rerouting for handling transient link and node failures," in *Proc. 24th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 4, Mar. 2005, pp. 2859–2863.

[214] K. Xi and H. Chao, "IP fast rerouting for single-link/node failure recovery," in *Proc. 4th Int. Conf. Broadband Commun. Netw. Syst. (BROADNETS)*, Sep. 2007, pp. 142–151.

[215] A. Kvalbein, A. F. Hansen, T. Čičic, S. Gjessing, and O. Lysne, "Multiple routing configurations for fast IP network recovery," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 473–486, Apr. 2009.

[216] I. Theiss and O. Lysne, "FROOTS—Fault handling in up*/down* routed networks with multiple roots," in *Proc. High Perform. Comput. HiPC*, 2003, pp. 106–117.

[217] D. Imahama, Y. Fukushima, and T. Yokohira, "A reroute method using multiple routing configurations for fast IP network recovery," in *Proc. 19th Asia–Pac. Conf. Commun. (APCC)*, Aug. 2013, pp. 433–438.

[218] T. A. Kumar and M. H. M. K. Prasad. (2012). *Enhanced Multiple Routing Configurations for Fast IP Network Recovery From Multiple Failures*. [Online]. Available: http://arxiv.org/abs/1212.0311

[219] T. Čičic *et al.*, "Relaxed multiple routing configurations: IP fast reroute for single and correlated failures," *IEEE Trans. Netw. Service Manag.*, vol. 6, no. 1, pp. 1–14, Mar. 2009.

[220] S. Cho, T. Elhourani, and S. Ramasubramanian, "Independent directed acyclic graphs for resilient multipath routing," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 153–162, Feb. 2012.

[221] M. Nagy, J. Tapolcai, and G. Rétvári, "Node virtualization for IP level resilience," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1250–1263, Jun. 2018.

[222] M. Menth and R. Martin, "Network resilience through multi-topology routing," in *Proc. 5th Int. Workshop Design Rel. Commun. Netw. (DRCN)*, 2005, pp. 271–277.

[223] T. Čičic, A. F. Hansen, A. Kvalbein, M. Hartman, R. Martin, and M. Menth, "Relaxed multiple routing configurations for IP fast reroute," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, 2008, pp. 457–464.

[224] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Multi-topology (MT) routing in OSPF," IETF, Fremont, CA, USA, RFC 4915, Jun. 2007. [Online]. Available: https://tools.ietf.org/html/rfc4915

[225] A. Atlas, K. Tiruveedhula, C. Bowers, J. Tantsura, and I. Wijnands, "LDP extensions to support maximally redundant trees," IETF, RFC 8320, Feb. 2018. [Online]. Available: http://tools.ietf.org/rfc/rfc8320

[226] A. Gopalan and S. Ramasubramanian, "Multipath routing and dual link failure recovery in IP networks using three link-independent trees," in *Proc. 5th IEEE Int. Conf. Adv. Telecommun. Syst. Netw. (ANTS)*, Dec. 2011, pp. 1–6.

[227] A. Gopalan and S. Ramasubramanian, "IP fast rerouting and disjoint multipath routing with three edge-independent spanning trees," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1336–1349, Jun. 2016.

[228] T. Elhourani, A. Gopalan, and S. Ramasubramanian, "IP fast rerouting for multi-link failures," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 3014–3025, Oct. 2016.

[229] E. Palmer, "On the spanning tree packing number of a graph: A survey," *Discr. Math.*, vol. 230, no. 1, pp. 13–21, 2001.

[230] M. Menth and W. Braun, "Performance comparison of not-via addresses and maximally redundant trees (MRTs)," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, 2013, pp. 218–225.

[231] *IP Routing: OSPF Configuration Guide, Cisco IOS Release 15.2s—OSPF IPv4 Remote Loop-Free Alternate IP Fast Reroute*, Cisco Syst., San Jose, CA, USA, Apr. 2012.

[232] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer, "IPv6 segment routing header (SRH)," IETF, Fremont, CA, USA, RFC 8754, Mar. 2020. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8754.html

[233] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," in *Proc. Conf. Appl. Technol. Architect. Protocols Comput. Commun. (SIGCOMM)*, 2000, pp. 175–187. [Online]. Available: https://doi.org/10.1145/347059.347428

[234] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating Internet routing instabilities," in *Proc. Conf. Appl. Technol. Architect. Protocols Comput. Commun. (SIGCOMM)*, 2004, pp. 205–218. [Online]. Available: https://doi.org/10.1145/1015467.1015491

[235] J. Chandrashekar, Z. Duan, Z. . Zhang, and J. Krasky, "Limiting path exploration in BGP," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 4, 2005, pp. 2337–2348.

[236] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, "R-BGP: Staying connected in a connected world," in *Proc. 4th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2007, p. 25.

[237] N. Gvozdiev, B. Karp, and M. Handley, "LOUP: The principles and practice of intra-domain route dissemination," in *Proc. 10th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2013, pp. 413–426.

[238] T. Holterbach, S. Vissicchio, A. Dainotti, and L. Vanbever, "SWIFT: Predictive fast reroute," in *Proc. Conf. ACM Special Interest Group Data Commun. (SIGCOMM)*, 2017, pp. 460–473. [Online]. Available: https://doi.org/10.1145/3098822.3098856

[239] T. Holterbach, E. C. Molero, M. Apostolaki, A. Dainotti, S. Vissicchio, and L. Vanbever, "BLINK: Fast connectivity recovery entirely in the data plane," in *Proc. 16th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 161–176.

[240] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek, "Measuring the effects of Internet path faults on reactive routing," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 126–137, Jun. 2003. [Online]. Available: https://doi.org/10.1145/885651.781043

[241] T. G. Griffin and B. J. Premore, "An experimental analysis of BGP convergence time," in *Proc. 9th Int. Conf. Netw. Protocols (ICNP)*, 2001, pp. 53–61.

[242] Z. M. Mao, R. Bush, T. G. Griffin, and M. Roughan, "BGP beacons," in *Proc. 3rd ACM SIGCOMM Conf. Internet Meas. (IMC)*, 2003, pp. 1–14. [Online]. Available: https://doi.org/10.1145/948205.948207

[243] T. G. Griffin and G. Wilfong, "On the correctness of IBGP configuration," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 17–29, Aug. 2002. [Online]. Available: https://doi.org/10.1145/964725.633028

[244] M. Caesar, L. Subramanian, and R. H. Katz, "Towards localizing root causes of BGP dynamics," EECS Dept., Univ. California at Berkeley, Berkeley, CA, USA, Rep. UCB/CSD-03-1292, 2003. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2003/6364.html

[245] U. Javed, I. Cunha, D. Choffnes, E. Katz-Bassett, T. Anderson, and A. Krishnamurthy, "PoiRoot: Investigating the root cause of interdomain path changes," in *Proc. ACM SIGCOMM Conf.*, 2013, pp. 183–194. [Online]. Available: https://doi.org/10.1145/2486001.2486036

[246] K.-K. Yap *et al.*, "Taking the Edge off with Espresso: Scale, reliability and programmability for global Internet peering," in *Proc. Conf. ACM Special Interest Group Data Commun. (SIGCOMM)*, 2017, pp. 432–445. [Online]. Available: https://doi.org/10.1145/3098822.3098854

[247] O. Bonaventure, C. Filsfils, and P. Francois, "Achieving sub-50 milliseconds recovery upon BGP peering link failures," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1123–1135, Oct. 2007. doi: 10.1109/TNET.2007.906045.

[248] M. Kopka. (2013). *IP Routing Fast Convergence*. [Online]. Available: https://www.cisco.com/c/dam/global/cs_cz/assets/ciscoconnect/2013/pdf/T-SP4-IP_Routing_Fast_Convergence-Miloslav_Kopka.pdf

[249] C. Filsfils. (2007). *BGP Convergence in Much Less Than a Second*. [Online]. Available: http://newnog.net/meetings/nanog40/presentations/ClarenceFilsfils-BGP.pdf

[250] A. Vahdat, D. Clark, and J. Rexford, "A purpose-built global network: Google's move to SDN: A discussion with Amin Vahdat, David Clark, and Jennifer Rexford," *Queue*, vol. 13, no. 8, pp. 100–125, Oct. 2015. [Online]. Available: https://doi.org/10.1145/2838344.2856460

[251] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "OpenFlow: Meeting carrier-grade recovery requirements," *Comput. Commun.*, vol. 36, no. 6, pp. 656–665, Mar. 2013. [Online]. Available: https://doi.org/10.1016/j.comcom.2012.09.011

[252] *OpenFlow Switch Specification, Version 1.1.0 Implemented (Wire Protocol 0x02), Open Networking Foundation*, ONF Standard TS-002, Feb. 2011. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.1.0.pdf

[253] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "OpenFlow-based segment protection in Ethernet networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 5, no. 9, pp. 1066–1075, Sep. 2013.

[254] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takacs, and P. Sköldström, "Scalable fault management for OpenFlow," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2012, pp. 6606–6610.

[255] R. M. Ramos, M. Martinello, and C. Esteve Rothenberg, "SlickFlow: Resilient source routing in data center networks unlocked by OpenFlow," in *Proc. 38th Annu. IEEE Conf. Local Comput. Netw.*, 2013, pp. 606–613.

[256] B. Stephens, A. L. Cox, and S. Rixner, "Plinko: Building provably resilient forwarding tables," in *Proc. 10th ACM Workshop Hot Topics Netw. (HotNets-XII)*, 2013, pp. 1–7. [Online]. Available: https://doi.org/10.1145/2535771.2535774

[257] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656890

[258] M. Borokhovich, L. Schiff, and S. Schmid, "Provable data plane connectivity with local fast failover: Introducing OpenFlow graph algorithms," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2014, pp. 121–126. [Online]. Available: https://doi.org/10.1145/2620728.2620746

[259] N. L. M. Van Adrichem, B. J. Van Asten, and F. A. Kuipers, "Fast recovery in software-defined networks," in *Proc. 3rd Eur. Workshop Softw. Defined Netw.*, 2014, pp. 61–66.

[260] C. Cascone, L. Pollini, D. Sanvito, A. Capone, and B. Sansó, "SPIDER: Fault resilient SDN pipeline with recovery delay guarantees," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, 2016, pp. 296–302.

[261] B. Stephens, A. L. Cox, and S. Rixner, "Scalable multi-failure fast failover via forwarding table compression," in *Proc. Symp. SDN Res. (SOSR)*, 2016, p. 12. [Online]. Available: https://doi.org/10.1145/2890955.2890957

[262] D. Merling, W. Braun, and M. Menth, "Efficient data plane protection for SDN," in *Proc. 4th IEEE Conf. Netw. Softw. Workshops (NetSoft)*, 2018, pp. 10–18.

[263] R. Sedar, M. Borokhovich, M. Chiesa, G. Antichi, and S. Schmid, "Supporting emerging applications with low-latency failover in P4," in *Proc. Workshop Netw. Emerg. Appl. Technol. (NEAT)*, 2018, pp. 52–57. [Online]. Available: https://doi.org/10.1145/3229574.3229580

[264] M. Menth, M. Schmidt, D. Reutter, R. Finze, S. Neuner, and T. Kleefass, "Resilient integration of distributed high-performance zones into the BelWue network using OpenFlow," *Comm. Mag.*, vol. 55, no. 4, pp. 94–99, Apr. 2017. [Online]. Available: https://doi.org/10.1109/MCOM.2017.1600177

[265] M. Reitblatt, M. Canini, A. Guha, and N. Foster, "FatTire: Declarative fault tolerance for software-defined networks," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2013, pp. 109–114. [Online]. Available: https://doi.org/10.1145/2491185.2491187

[266] Barefoot. (Sep. 2020). *Tofino: World's Fastest P4 Programmable Ethernet Switch ASIC*. [Online]. Available: https://www.barefootnetworks.com/products/brief-tofino/

[267] E. Gafni and D. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Trans. Commun.*, vol. COM-29, no. 1, pp. 11–18, Jan. 1981.

[268] M. Chiesa *et al.*, "PURR: A primitive for reconfigurable fast reroute: Hope for the best and program for the worst," in *Proc. 15th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2019, pp. 1–14. [Online]. Available: https://doi.org/10.1145/3359989.3365410

[269] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A fault-tolerant engineered network," in *Proc. 10th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2013, pp. 399–412.

[270] Y. Pignolet, S. Schmid, and G. Tredan, "Load-optimal local fast rerouting for resilient networks," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, 2017, pp. 345–356.

[271] M. Borokhovich, Y. Pignolet, S. Schmid, and G. Tredan, "Load-optimal local fast rerouting for dense networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2583–2597, Dec. 2018.

[272] K. Foerster, A. Kamisiński, Y. Pignolet, S. Schmid, and G. Tredan, "Bonsai: Efficient fast failover routing using small arborescences," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, 2019, pp. 276–288.

[273] K. Foerster, A. Kamisiński, Y. Pignolet, S. Schmid, and G. Tredan, "Improved fast rerouting using postprocessing," in *Proc. 38th Symp. Rel. Distrib. Syst. (SRDS)*, 2019, Art. no. 17309.

[274] M. Borokhovich and S. Schmid, "How (not) to shoot in your foot with SDN local fast failover," in *Principles of Distributed Systems*, R. Baldoni, N. Nisse, and M. van Steen, Eds. Cham, Switzerland: Springer Int., 2013, pp. 68–82.

[275] M. Chiesa, I. Nikolaevskiy, A. Panda, A. V. Gurtov, M. Schapira, and S. Shenker. (2014). *Exploring the Limits of Static Failover Routing*. [Online]. Available: http://arxiv.org/abs/1409.0034

[276] K. Foerster, Y. Pignolet, S. Schmid, and G. Tredan, "CASA: Congestion and stretch aware static fast rerouting," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2019, pp. 469–477.

[277] H. Villför, "Operator experience from ISIS convergence tuning," presented at the RIPE 47 Meeting, 2004. [Online]. Available: https://meetings.ripe.net/ripe-47/presentations/ripe47-routing-isis.pdf

[278] P. Francois, M. Shand, and O. Bonaventure, "Disruption free topology reconfiguration in OSPF networks," in *Proc. IEEE INFOCOM 26th Int. Conf. Comput. Commun.*, 2007, pp. 89–97.

[279] M. Shand, S. Bryant, S. Previdi, C. Filsfils, P. Francois, and O. Bonaventure, "Framework for loop-free convergence using the ordered forwarding information base (oFIB) approach," IETF, Fremont, CA, USA, RFC 6976, Jul. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6976

[280] K. Foerster, S. Schmid, and S. Vissicchio, "Survey of consistent software-defined network updates," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1435–1461, 2nd Quart., 2019.

[281] M. Markovitch and S. Schmid, "SHEAR: A highly available and flexible network architecture: Marrying distributed and logically centralized control planes," in *Proc. 23rd IEEE Int. Conf. Netw. Protocols (ICNP)*, 2015, pp. 1–9.

[282] S. Vissicchio, L. Cittadini, O. Bonaventure, G. G. Xie, and L. Vanbever, "On the co-existence of distributed and centralized routing control-planes," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 469–477.

[283] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 756–767, May 2002.

[284] Cisco Systems. (2020). *Cisco ASR 901S Series Aggregation Services Router Software Configuration Guide*. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/wireless/asr_901s/scg/b_scg_for_asr901s/b_scg_for_asr901s_chapter_0100100.html

[285] J. Liu, B. Yan, S. Shenker, and M. Schapira, "Data-driven network connectivity," in *Proc. 10th ACM Workshop Hot Topics Netw. (HotNets)*, Nov. 2011, pp. 1–6.

[286] W. Braun, M. Albert, T. Eckert, and M. Menth, "Performance comparison of resilience mechanisms for stateless multicast using BIER," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, 2017, pp. 230–238.

[287] D. Merling, S. Lindner, and M. Menth, "Comparison of fast-reroute mechanisms for BIER-based IP multicast," in *Proc. Int. Conf. Softw. Defined Syst.*, 2020, pp. 1–8.

[288] Z. Zhang and A. Baban, "Bit index explicit replication (BIER) forwarding for network device components," U.S. Patent 9 705 784, Jul. 11, 2017.

Marco Chiesa received the Ph.D. degree in computer engineering from Roma Tre University in 2014. He is an Assistant Professor with the KTH Royal Institute of Technology, Sweden. His research interests include Internet architectures and protocols, including aspects of network design, optimization, security, and privacy. He received the IEEE William R. Bennett Prize in 2020, the IEEE ICNP Best Paper Award in 2013, and the IETF Applied Network Research Prize in 2012. He has been a Distinguished TPC Member at IEEE Infocom in 2019 and 2020.

Andrzej Kamisiński received the B.Sc., M.Sc., and Ph.D. degrees from the AGH University of Science and Technology, Kraków, Poland, in 2012, 2013, and 2017, respectively, where he is an Assistant Professor. In 2015, he joined the QUAM Lab, NTNU, Trondheim, Norway, where he worked with Prof. B. E. Helvik and with Telenor Research on dependability of software-defined networks. In summer 2018, he was a Visiting Research Fellow with the Communication Technologies Group led by Prof. S. Schmid with the Faculty of Computer Science, University of Vienna, Austria. From 2018 and 2020, he was a member of the Management Committee of the Resilient Communication Services Protecting End-User Applications From Disaster-Based Failures European COST Action, and a Research Associate with the Networked Systems Research Laboratory, School of Computing Science, University of Glasgow, Scotland, in 2020. His primary research interests span dependability and security of computer and communication networks.

Jacek Rak (Senior Member, IEEE) received the M.Sc., Ph.D., and D.Sc. (Habilitation) degrees from the Gdańsk University of Technology, Gdańsk, Poland, in 2003, 2009, and 2016, respectively, where he is an Associate Professor and the Head of the Department of Computer Communications. He has authored over 100 publications, including the book *Resilient Routing in Communication Networks* (Springer, 2015). From 2016 and 2020, he was leading the COST CA15127 Action *Resilient Communication Services Protecting End-User Applications From Disaster-Based Failures* (RECODIS) involving over 170 members from 31 countries. His main research interests include the resilience of communication networks and networked systems. He has also served as a TPC member of numerous conferences and journals. He has been the General Chair of ITS-T 2017 and MMM-ACNS 2017, the General Co-Chair of NETWORKS 2016, the TPC Chair of ONDM 2017, and the TPC Co-Chair of IFIP Networking 2019. He is the Member of the Editorial Board of *Optical Switching and Networking* (Elsevier) and the Founder of the International Workshop on Resilient Networks Design and Modeling.

Gábor Rétvári received the M.Sc. and Ph.D. degrees in electrical engineering from the Budapest University of Technology and Economics in 1999 and 2007, respectively, where he is currently a Senior Research Fellow with the Department of Telecommunications and Media Informatics. His research interests include all aspects of network routing and switching, the programmable data plane, and the networking applications of computational geometry and information theory. He maintains several open source scientific tools written in Perl, C, mand Haskell.

Stefan Schmid received the M.Sc. and Ph.D. degrees from ETH Zurich, Switzerland, in 2004 and 2008, respectively. He is a Professor with the Faculty of Computer Science, University of Vienna, Austria. He was a Postdoctoral Researcher with TU Munich and the University of Paderborn in 2009, a Senior Research Scientist with the Telekom Innovations Laboratories (T-Labs), Berlin, Germany, from 2009 to 2015, and an Associate Professor with Aalborg University, Denmark, from 2015 to 2018. He is currently leading the ERC Consolidator Project *AdjustNet* and the WWTF Project WhatIf.