electronics

Review

A Comprehensive Survey of In-Band Control in SDN: Challenges and Opportunities

David Carrascal, Elisa Rojas *,†, Jose M. Arco †, Diego Lopez-Pajares and Joaquin Alvarez-Horcajo and Juan Antonio Carral

Departamento de Automática, Universidad de Alcalá, Escuela Politécnica Superior, 28805 Alcalá de Henares, Spain
* Correspondence: elisa.rojas@uah.es
† These authors contributed equally to this work.

Abstract: Software-Defined Networking (SDN) is a thriving networking architecture that has gained popularity in recent years, particularly as an enabling technology to foster paradigms like edge computing. SDN separates the control and data planes, which are later on synchronised via a control protocol such as OpenFlow. In-band control is a type of SDN control plane deployment in which the control and data planes share the same physical network. It poses several challenges, such as security vulnerabilities, network congestion, or data loss. Nevertheless, despite these challenges, in-band control also presents significant opportunities, including improved network flexibility and programmability, reduced costs, and increased reliability. Benefiting from the previous advantages, diverse in-band control designs exist in the literature, with the objective of improving the operation of SDN networks. This paper surveys the different approaches that have been proposed so far towards the advance in in-band SDN control, based on four main categories: automatic routing, fast failure recovery, network bootstrapping, and distributed control. Across these categories, detailed summary tables and comparisons are presented, followed by a discussion on current trends a challenges in the field. Our conclusion is that the use of in-band control in SDN networks is expected to drive innovation and growth in the networking industry, but efforts for holistic and full-fledged proposals are still needed.

Keywords: Software-Defined Networking; in-band control; survey; network bootstrapping; fault recovery; routing; distributed control

check for updates

Citation: Carrascal, D.; Rojas, E.; Arco, J.M.; Lopez-Pajares, D.; Alvarez-Horcajo, J.; Carral, J.A. A Comprehensive Survey of In-Band Control in SDN: Challenges and Opportunities. *Electronics* **2023**, *12*, 1265. https://doi.org/10.3390/electronics12061265

Academic Editor: Christos J. Bouras

Received: 31 January 2023
Revised: 21 February 2023
Accepted: 5 March 2023
Published: 7 March 2023

1. Introduction

During the last decade, the flourishing of SDN [1] has led to a wide range of novel network services and applications, including inter-disciplinary use cases, such as those envisioned by the fifth generation of mobile technologies (5G) and beyond [2]. The cornerstone of SDN is the possibility to separate the control and data planes, hence providing multiple advantages in network management and operational costs, among others.

Uncoupling both planes involves the utilization of a control communication protocol that is in charge of synchronising both planes. Considering the SDN architecture, this protocol is part of the Southbound Interface (SBI), formally known as Data-Controller Plane Interface (D-CPI), as defined by the Open Networking Foundation (ONF) [3]. Although the control protocol that initiated the SDN paradigm is OpenFlow [4], others currently exist, such as P4Runtime [5]; legacy ones (e.g., Simple Network Management Protocol (SNMP)) are even considered potential control protocols in the generalised SDN architectural model.

When deploying the control channel, it is possible to follow either an out-of-band or in-band approach, as depicted in Figure 1. On the one hand, the channel is out-of-band when the physical links leveraged for communication are different for the control and data planes, as illustrated in Figure 1a. In this scenario, the network requires at least one extra physical link per SDN device, apart from the data plane topology, for the control channel. On the other hand, the channel is in-band when physical links are shared between the control and data planes and no additional links are required. Additionally, hybrid-band control is defined when some links are shared but, at the same time, dedicated links for control also exist [6], as in the example shown in Figure 2.

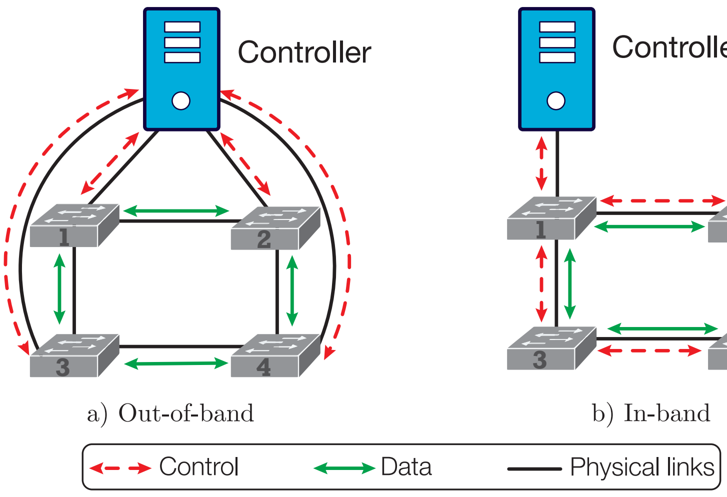

Figure 1. SDN out-of-band and in-band control networks.

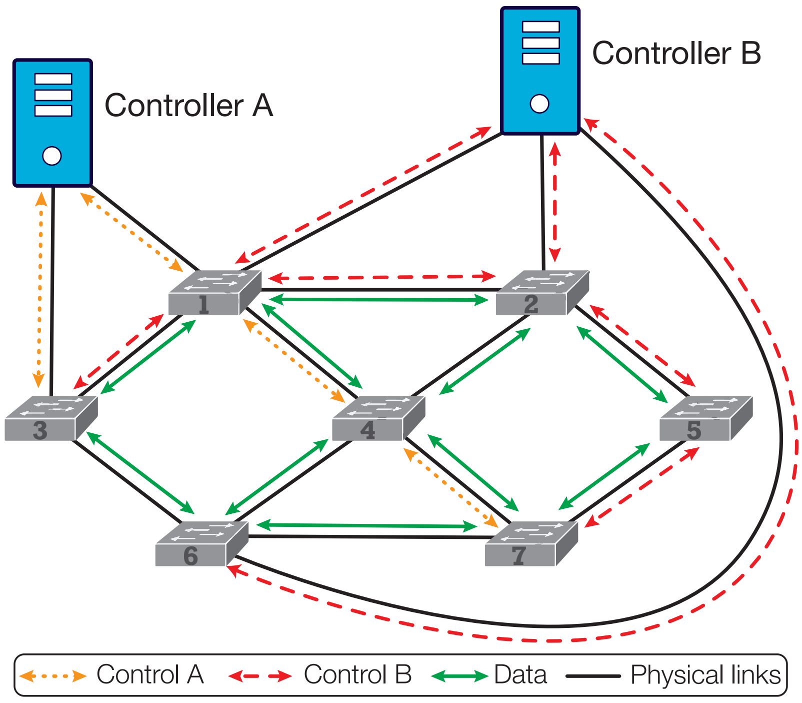

Figure 2. SDN hybrid-band control network.

Each type of deployments has its own advantages [6], and the selected approach mainly depends on the network scenario [7–9]. Some of the disadvantages of out-of-band are advantages of in-band and vice versa, while hybrid-band could be considered a trade-off among both approaches. We summarised the main features of each model in Table 1:

Table 1. Features of in-band and out-of-band control.

Feature	Out-of-Band Control	In-Band Control
SDN device configuration	Simple	Complex
Control channel security	Safe, isolated channel	Risky, shared channel
Network maintenance and deployment costs	High	Low
Scalability	Poor	Good
Resilience	Costly (re-wire)	Fast failure recovery (easy traffic re-routing)

The current trend shows that in-band seems to be prominently used in SDN deployments [10], particularly for large networks in which the cost of using out-of-band might be exorbitant. Additionally, in-band control allows a wide range of new applications, particularly for constrained and hybrid SDN environments [11,12], in which out-of-band control is complex (or even impossible) to deploy. Some of the use cases leveraging in-band control include 5G [13] and Non-Terrestrial Networks (NTNs) [14], Internet of Things (IoT) (e.g., underwater networks [15], energy saving [16] or resource-constrained environments [17]), emergency and disaster situations [18,19], and enhanced network security [20].

Nevertheless, despite of the multiple benefits of in-band control in SDN scenarios, few efforts have been made to design a common and holistic in-band control protocol. This design should consider standard and popular platforms (both for SDN controllers and switches/devices) for a complete integration in current SDN deployments. Therefore, the aim of this survey is to provide the current state of the art in this area, describing current proposals, technologies used, advantages and disadvantages, etc., so that it can serve as a comprehensive report for researchers and network managers interested in this topic. Therefore, our goal is to provide a comprehensive survey of the topic and a comprehensive overview for academia and industry. Accordingly, the main contributions of our survey are

- A comprehensive list of works about in-band SDN control proposals.
- A classification of the surveyed works in terms of designed features, defining four categories.
- A chronological table per category stating achieved functionality and maturity of each proposal.
- A summarised analysis for each proposal, describing its main characteristics, implementation details, and results, if applicable.
- A final discussion and comparison of all works, stating potential research trends and future challenges in the field.

After providing an overview about the motivation and objectives of this survey in Section 1, the remainder of the manuscript is structured as follows. In Section 2, we review the related work (including other SDN surveys, and technologies and platforms associated with in-band SDN control) and summarise the contributions of this survey. In Section 3, we describe the methodology followed to look for related articles and current statistics about the topic. The main survey is performed in Section 4, in which we cover all previously found articles, first providing a classification. Afterwards, Section 5 discusses current research trends and future challenges according to the performed analysis. Finally, Section 6 concludes the survey.

2. Related Work and Contribution of This Survey

After reviewing the literature, there are several surveys on SDN. To begin, there are various general surveys with a high number of citations [1,21–29]: all these surveys have in common that they were published at the same time as the SDN wave. They cover the challenges and new aspects that SDN brought and how it shattered the conventional networking paradigm.

As a technology becomes trendy in a field, challenges, opportunities and threats emerge. Thus, as SDN networks emerged, so did works [30–33] that addressed the security issues of such networks. One very controversial aspect is the centralisation of all network control in a single entity. In these works, they indicate lines of work, challenges, risks, threats, and works that propose countermeasures to deal with the most common threats.

As soon as the new technology stops being trendy and starts to be implemented in the industry, it is necessary to start studying how to make the transition from the old paradigm to the SDN paradigm. It is therefore necessary to study and analyse the necessary intermediate steps, as it is impossible to replace all legacy equipment in a single step. This is where hybrid SDN architectures and the corresponding surveys where they are studied in detail come in [11,34,35].

Another aspect that has been addressed [36,37] in the SDN field is distributed systems where more than one control entity come into play to manage a group of SDN nodes. These systems solve many of the performance problems, bottlenecks, and threats that arise from centralisation, but they also bring with them challenges in managing shared resources, such as race conditions, and atomicity in instantiating rules in SDN nodes, among others. Finally, another topic that has been deeply explored [38,39] is the integration of other trendy technologies such as IoT with SDN, what challenges arise in the integration, and what benefits the IoT environments might derive from the SDN paradigm. Additionally, other specialised surveys analyse SDN programming languages [40], link failure recovery [41], or Quality of Service (QoS) [42].

The aforementioned works represent only a small part of the total number of surveys on SDN. Our intention is to highlight some of the most referenced works and the topics they cover, while emphasising that an analysis of all existing surveys on SDN is beyond the scope of this article due to their large number. All the publications mentioned above have in common that they do not cover the in-band control paradigm in depth. For this reason, this review is of particular interest as it brings together all the advances in this area in a single work. Nevertheless, to date and to the best of our knowledge, there has been no survey of in-band control in SDN, which is remarkable given its importance for real SDN deployments.

Finally, we would like to clarify that in-band control is not the same (and not even directly related) to in-band network telemetry [43]. Although the latter also emerged with the growth of SDN and programmable networks, in-band network telemetry is a technique that uses the data plane to directly drive the network measurement process, while in-band control is a model to deploy control channels leveraging the existing data plane channels. In summary, in-band telemetry is related with data plane devices (hardware), while in-band control is related with data plane channels/link (logic).

3. Survey Methodology and Statistics

In order to comprehensively survey the topic of in-band control in SDN, we performed a manual search (in which all works were checked one by one) using Google Scholar. The core keywords used for this search were: *in band*, *in-band*, *SDN*, *Software-Defined Networking*, *Software-Defined Network*, *softwarized network*, *softwarization*, *control*, *communication*, and *survey*. In all performed searches, all works were manually inspected (mainly by reviewing title and abstract), from most to less relevant (according to the default Google Scholar resultant list), until the search provided around 10 consecutive non-related works: we considered that a specific search would not produce more related results. Only works in English and Spanish (these were the languages that the authors fully understand) were considered.

The first lookup was executed in August 2021 and periodically repeated approximately every 3 months until the last one was performed at the end of January 2023, when this survey was finally sent for revision. Additionally, the authors subscribed to the topics *sdn survey "in band"-"intitle:telemetry"* in Google Scholar to obtain alerts about newly published topics in the field. As it can be seen, the keyword *telemetry* was avoided in our searches because it is not related to our survey, as mentioned in Section 2.

Once the survey was finalised, in January 2023, we performed an additional search of keywords to have a quick overview of the current trends in the field. This search was automatic this time, looking for the keywords *SDN in-band* in Google Scholar once again and counting the amount of published articles. The result is depicted in Figure 3, in which the blue line (with circles) represents the amount of works found in Google Scholar for those specific years. As it can be observed, the trend rapidly grows from 2012 to 2018 and stays stable in the later years. However, it is important to highlight that this is an automatic search and, as such, it might include as well works for in-band telemetry in some cases (as they were not explicitly excluded). In any case, it provides a clear overview of the trend.

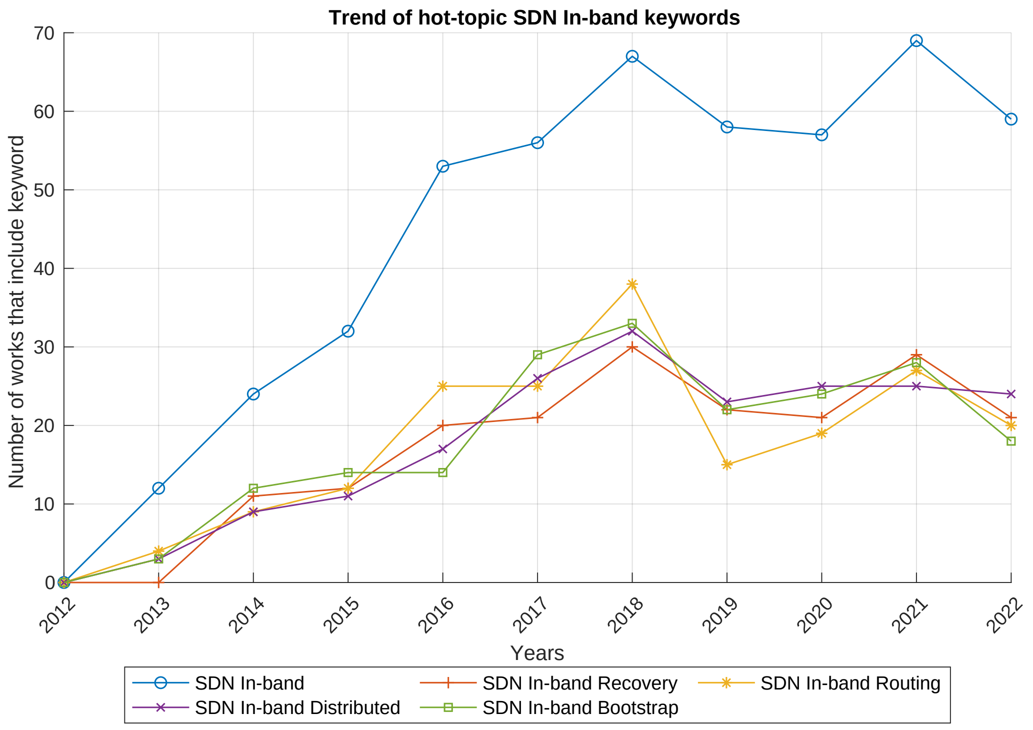

Figure 3. Trend of published articles related to SDN in-band.

Additionally, based on the previously examined works, which were classified into four main topics (as detailed in Section 4, four additional searches were performed as well including those classification items: *SDN in-band recovery*, *SDN in-band routing*, *SDN in-band distributed*, and *SDN in-band bootstrap*. As illustrated in Figure 3, each of them covers almost half of the total articles in the automatic search.

Finally, we measured the importance of the topic as the number of references each article had obtained. More specifically, Figure 4 depicts how many papers have a number of citations in the ranges 0, 1–10, 11–20, etc. As it can be observed, the majority of works have received few citations, between 1 and 10, or none at all.

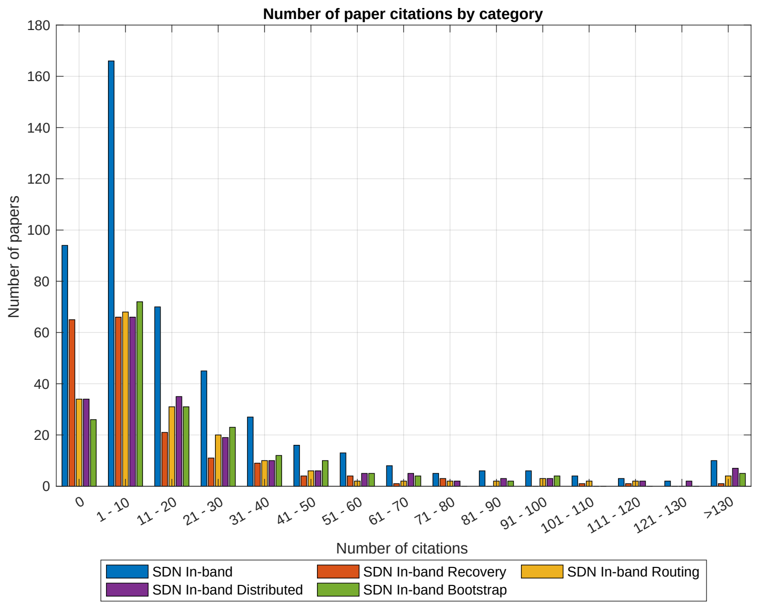

Figure 4. Number of papers related to SDN in-band and their amount of citations.

4. A Survey of In-Band SDN Control in SDN

In this section, we survey all works that were classified into four main categories, which can be found in Sections 4.2–4.5. Prior to this description, Section 4.1 provides a preliminary overview of the survey, describing those four classification items and a final subsection focusing on the technologies and tools commonly used in SDN. Finally, in Section 4.6, other works that do not directly implement in-band control, but are closely related, are examined.

4.1. Preliminary Survey Overview and Classification

Before studying each of the surveyed works, we decided first to provide an overview with patterns and similarities found in the articles.

Among the motivations for each design, we discovered the main following ones, which are illustrated in Figure 5 as four categories:

- Automatic routing;
- Fast failure recovery;
- Network bootstrapping;
- Distributed control.

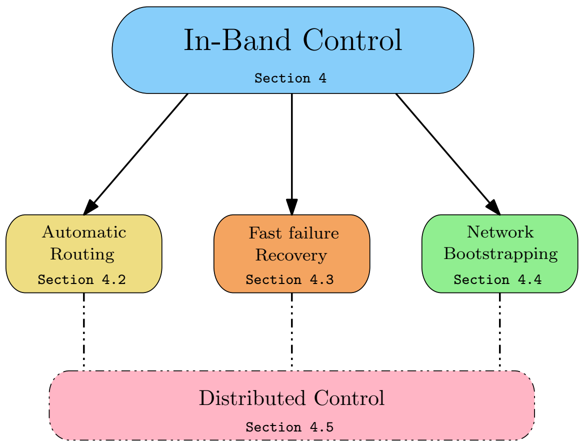

Figure 5. Main categories of the survey.

The three first categories could be seen as three main features that in-band control should provide, while the fourth is an orthogonal aspect. More specifically, the need for **automatic routing** in in-band control seems quite straightforward: while out-of-band control is usually implemented with direct links towards the controller (although it should not be necessarily like that), in-band control requires calculation of routes between SDN devices and the controller/s and vice versa. **Fast failure recovery** could be considered one of the main advantages of in-band; by simply selecting a different route in the data plane, the control communication can be recovered after a link or switch failure. However, the procedure to build these new routes should ideally be defined in accordance with the routing mechanism. Regarding **network bootstrapping**, it refers to the ability of setting up all required parameters (such as the connectivity details for the controller–switch communication) prior to startup of the system. Network bootstrapping is also a desirable feature for out-of-band control, but it is almost a must for in-band, since routes might vary depending on each deployment and the routing method applied. Finally, if the network has **distributed control** (or controllers), it implies that two or more controllers are involved in the control decisions and, therefore, a coordination mechanism should be defined for routing, failure recovery, and bootstrapping. Additionally, in-band control can also serve as a potential means for internal communication among controllers.

The previous four aspects were the most commonly found. For that reason, they were selected as the four main classification items of our survey. Nevertheless, some other features were also found in the literature, such as **topology discovery**, the possibility of combining both out-of-band and in-band as a **hybrid-band** deployment, and other design parameters such as scalability or QoS. Although network topology discovery is a key element for a number of necessary network functionalities, it was decided to not include it as a key aspect to be analysed in the survey. This is because, in general, it is usually a process that is independent of the in-band protocols, as some solutions include a topology discovery system/function, while others rely on independent topology discovery protocols that are used to obtain the necessary paths to compute the in-band routes. To delve into these aspects, let us consider Figure 2 again. The network represented in the figure contains seven SDN switches and two SDN controllers. The controllers are distributed and control type is hybrid-band, as both types are combined. For example, Controller A is connected to switches 1 and 3 via out-of-band control, but also to 4 and 7 via in-band control. The case of Controller B is similar, with some switches connected via out-of-band control, other ones via in-band; there are even some combining both option, such as switches 1 and 2, which can connect to Controller B using both types of control.

In this scenario, fast failure recovery can be implemented via restoration or protection (as defined by Sharma et al. [44]). For instance, switch 1 could be connected to Controller A and Controller B at the same time via out-of-band control, while maintaining a third in-band control route to Controller B routed through switch 2 for enhanced resiliency. Furthermore, bootstrapping could help to deploy all initial parameters and default setup and this might require topology discovery for it. Although it might seem that routing and topology discovery are both closely related, not all routing proposals might need to obtain the full topology; this is especially true if it is based on tree routing. For example, in Figure 2, all switches have at least one route towards a controller, but link 2–4, 3–6, 4–6, and 6–7 might be completely unknown for the control plane if a topology discovery function is not implemented.

Common Technologies and Tools

It is also important to highlight the most common technologies and tools found in the surveyed works in order to provide an overview of the protocols, controllers, switches and simulation, and emulation tools currently used in in-band SDN environments.

First of all, almost all proposals in the D-CPI/SBI use OpenFlow [4] protocol, while some of the most recent ones consider Programming Protocol-independent Packet Processors (P4) [45,46] and/or P4Runtime [5]. OpenFlow exchanges messages between the controller and its legacy switches to communicate the control and data planes. The controller indicates forwarding or metric rules to the switches by matching header fields. However, the reduced number of predefined OpenFlow rules limits its flexibility. This has led to new solutions such as P4 and P4Runtime, which create a new programmable data plane paradigm through a universal high-level language. It also replaces control and data plane communication with message exchange using Google Remote Procedure Call (gRPC), providing a fast and flexible communication interface.

In the case of SDN controllers, the most commonly leveraged ones are Open Network Operating System (ONOS) [47], OpenDaylight (ODL) [48], Ryu [49], and Floodlight, among others [1]. ONOS and ODL are high-performance controllers used in real-world deployments, while Ryu and Floodlight are simpler and intended for prototyping or low-performance use. However, there are also several proposals that simply simulate the behaviour of the controller instead of using a real one.

Regarding SDN devices/switches, the most common by far is Open vSwitch (OVS) [50]; however, Basic OpenFlow Userspace Software Switch (BOFUSS) [51] is also used in a few works. The target, Behavioral Model version 2 (BMv2) [52], is also used, particularly when P4 is considered. In most cases, these switches were deployed using the Mininet [53] emulation platform. As for simulated environments, OMNeT++ [54,55] was predominant, followed by ns-3 [56,57]; many others exist as well [58].

Finally, for the evaluation, diverse types of topologies have been considered, including simple linear, star or mesh networks, data center networks like fat tree or Clos, or other modeled networks (adjusted by a set of parameters) like Waxman [59] or Barabási–Albert [60].

According to this preliminary study, we have organised all works in four main subsections, and for each of them, all surveyed items have been summarised in a table. This table contains the year the article was published, its key features, evaluation tools, if its code is publicly available, and an overall contribution based on three stars. The contribution was qualitatively assigned based on all previous characteristics together with its impact in number of citations.

4.2. Automatic Routing

To start, we describe the different articles found in the literature that propose designs for automatic routing in in-band control in SDN. Although the majority of surveyed works provide a routing approach, some of them only focused on routing (and failure recovery in some cases). Those are the ones classified in this initial section. To provide a quick overview, Table 2 summarises them all. Afterwards, we individually examine each work.

Sharma et al. [44] present the basic ideas for the design of in-band control in OpenFlow networks. They analyse the extent to which diverse SDN switches could support this type of control and implement a prototype with NOX and TrafficLab1.3 (currently known as BOFUSS [51]), tested with diverse topologies. Additionally, they examine how failure recovery is a cornerstone in this type of control, presenting the restoration and protection methods to implement it. This article is particularly relevant in the field, as it defined the foundations for automatic and resilient bootstrapping for in-band control in SDN.

Goltsmann et al. [61] present the Intermediate System to Controller (ICS) protocol, focused on resilient connectivity, while keeping protocol overhead low, for in-band control in SDN. The design relies on the fact that communication only occurs between the controller and the rest of nodes—not among all pairs of nodes. Considering this aspect, they design a labelled spanning tree and its behavior when a single link failure occurs. However, it requires some convergence time and multiple failures are not considered.

Khakhalin et al. [62] formulate an algorithm for automatic in-band control routing which is resilient to multiple failures. However, it requires the assignment of a unique label to every switch. This becomes quite complex when multiple failures occurs because loops should be avoided without a global knowledge of paths. Moreover, no reference implementation with a SDN controller is provided.

Table 2. Summary table of in-band proposals that support automatic routing.

Article	Year	Key Features	Evaluation & Tools	Available Code?	Overall Contribution
Sharma et al. [44]	2016	Key ideas for in-band control with DHCP, queueing and failure recovery	Emulation: NOX/BOFUSS/Pan-European, ring, star networks	—	★★★
Goltsmann et al. [61]	2017	Routing and single-link failure recovery	Analytical model	—	★☆☆
Khakhalin et al. [62]	2017	Routing and link failure recovery	Analytical model	—	★⯨☆
Raza et al. [63]	2017	Routing and link failure recovery	Analytical model/Random networks	—	★⯨☆
Gonzalez et al. [64]	2018	Routing and link failure recovery	Emulation: Ryu/OVS/Mesh network	—	★★☆
Mohan et al. [65]	2018	Routing and link failure recovery with a security focus	Analytical model/Internet topologies	—	★⯨☆
Görkemli et al. [66]	2018	Routing and link failure recovery	Emulation: POX/OVS/Ring-tree topology	—	★★⯨
Holzmann et al. [67]	2019	Distributed routing protocol for in-band SDN control	Simulation: OMNeT++/WAN/Zoo topology	—	★★☆
Raza et al. [68]	2019	Path selection algorithm for routing using MPTCP	Simulation: C++/BRITE networks	—	★☆☆
Fan et al. [69]	2020	Routing, in-band connection with QoS	Simulation: Python Networkx. Emulation: ONOS/OVS/Docker/Fat tree and random networks	—	★★⯨
Ningyuan et al. [70]	2021	Routing and link failure recovery in LEO networks	Simulation: Walker–Delta constellations	—	★★☆
Kumazoe et al. [71]	2022	Routing in P4BMv2-based network	Real testbed: BMv2/Linear network	—	★★☆

Raza et al. [63] focus on the in-band channel availability and propose the use of MultiPath TCP (MPTCP) instead of simply Transmission Control Protocol (TCP). High availability is achieved using as many disjoint paths to a switch as possible by selecting a set of adjacent switches to the controller. The search of this adjacent switches might be time consuming, so they only propose and evaluate two simple heuristic algorithms.

Gonzalez et al. [64] design an in-band OpenFlow channel through MPTCP which leverages the multiple paths for enhanced resilience. However, it requires an out-of-band channel for the initial setup.

Mohan et al. [65] propose a control path routing approach that selects two node-disjoint control paths for every switch in the network. In that way, a malicious node can be detected based on the normal packet-in messages sent on the control paths. Then, the controller can send explicit test flows to determine and isolate the malicious switch. They derive an optimization formula to provide a routing solution that minimises the average number of intermediate nodes while satisfying the malicious switch detection and resilience constraints. They demonstrate the effectiveness of the proposed approach through numerical analysis. Their results show that the proposed approach enables faster malicious switch detection with less control overhead compared to a previous approach. However, the solution does not scale for large networks due to the high number of decision variables.

Görkemli et al. [66] define a dynamic control plane architecture to redistribute and scale controller load and re-route in-band control traffic using control flow tables depending on underlying the network topology and the requirements of the applications running on top. They test their dynamic paradigm on a virtualised network with an arbitrary number of controllers (running a modified version of POX) and OVS switches in hybrid ring-tree and balanced binary tree topologies. They achieve greater efficiency in the use of the controller's resources than with a static approach.

Holzmann et al. [67] present Izzy, a distributed routing protocol for maintaining robust connectivity of in-band SDN control channels. Izzy is based on a combination of a spanning tree and topology-dependent temporary addresses called *locators*. The solution is divided into three modules; the first one, called TREE, is responsible for maintaining the spanning tree structure by assigning locators. The second module, known as UPDATER, is responsible for coordinating and updating the network's forwarding tables. Finally, the Fast ReRoute (FRR) module is in charge of maintaining backup routes. The evaluation of Izzy has been carried out in OMNeT++ and shows recovery times under 100 ms in WAN topologies of 100 nodes.

Raza et al. [68] define an algorithm to create robust control paths when leveraging MPTCP for in-band communication in SDN. Their algorithm looks for the best set of switches to route through the shortest and disjoint paths in the network. They compute the sum of disjoint path lengths and computation time and compare their proposal with two baseline algorithms, proving better results. However, they only test it via simulation and leave the real implementation as future work.

Fan et al. [69] propose a centralised routing solution to improve the QoS of in-band control traffic in terms of bandwidth. They analyse the in-band control traffic, finding out that it is periodic and needs a very small bandwidth: around 120 kbits every 5 s. Hence, they place the in-band connection on the path with the most available bandwidth. They design a centralised routing algorithm in the SDN controller to compute the optimal path from the network topology and current consumed bandwidth at every link. However, their simulations show that it does not scale to generic large networks. They choose to stick to a Fat Tree Data Center (DC) network and a special network designed with higher connectivity and randomness, in which they clearly improve the Round-Trip Time (RTT) and the Packet Loss Ratio (PLR) compared with a L2 Spanning Tree Protocol (STP).

Ningyuan et al. [70] acknowledge the need for reliable in-band control in Low Earth Orbit (LEO) constellation systems and propose a dual-layer architecture where the upper-layer control plane provides reliable in-band control paths for the lower-SDN control plane. This plane, in turn, deploys virtual out-of-band control in the data plane, in spite of the constantly changing of network topology. In order to provide reliable control traffic transmission, a redundant path-planning algorithm is further proposed. They evaluate the framework with a simulation which proves much better satellites accessibility and has barely any inaccessible satellites under massive link faults.

Kumazoe et al. [71] design an in-band control channel communication scheme for P4, in which control messages are piggybacked onto data packets. The target switch is based on the BMv2, and their evaluation shows performance degradation in data forwarding, which they envision to be solved as future research work.

4.3. Fast Failure Recovery

This section covers a set of surveyed works exclusively focused on fast failure recovery methods for in-band control in SDN. These proposals might complement the rest: particularly, those that did not envision failure recovery in the first place. Table 3 summarises all works in this section, which are described in detail in the remainder of this section.

Table 3. Summary table of in-band proposals that support fast failure recovery.

Article	**Year**	**Key Features**	**Evaluation & Tools**	**Available Code?**	**Overall Contribution**
Huang et al. [72]	2016	Protection method for link failure recovery	Simulation: Python	—	★⯪☆
Park et al. [73]	2017	Restoration method for link failure recovery	Real testbed: ONOS/Raspberry Pi, OVS/Mesh network	—	★★☆
Akhtar et al. [74]	2018	Mechanism to avoid retransmitions of in-band packets	Emulation: Ryu/Mininet/Linear network	—	★⯪☆
An et al. [75]	2019	Protection method for interferences in in-band wireless networks	Real testbed: ONOS/Raspberry Pi/Mesh network (IEEE 802.11)	—	★★☆
Alowa et al. [76]	2020	Protection method for link failure recovery	Simulation: C++/SNDlib networks	—	★☆☆
Ochoa-Aday et al. [77]	2020	Fast multiple-link failure recovery	Simulation: OMNeT++/Atlanta, Sun, Pioro networks	—	★★☆

Huang et al. [72] design a protection method for failures in the SDN control channel in Heterogeneous Networks (HetNets), usually implemented as in-band. Under a link failure, existing approaches rely on local rerouting that can bring congestion in the neighbouring links. They propose a weighted cost-minimization problem, where the traffic load balancing and path cost are jointly considered. This is an NP-hard problem and their solution is a near-optimal Markov approximation. After simulation in Python with diverse network types, number of controllers, and multiple failures, it proves to have better results (in resource utilisation and recovery time) than other standard routing algorithms.

Park et al. [73] define an algorithm for fast in-band control channel recovery after link failure. On receiving a port down message from the failure affected switch, the controller finds the shortest paths to the affected switch and searches the target switches (*Tswitch*) where the flow addition or modification is required. Then, the controller tests the shortest paths, transmitting 1000 flows on each paths. After that, it sends an special check message (request-reply) to each "Tswitch" switch to measure response time. Finally, it selects the path with the smallest delay. They compare it with other conventional detour approaches and it reduces both the recovery time and the detour traffic used for it.

Akhtar et al. [74] focus their proposal in mitigating the effects of delay and losses on the control plane traffic. They propose to use Network Coding (NC) on control flows to reduce both the number of transmissions in case of packet losses and the delay incurred. A NC switch combines (codes) packets to the same destination in a new packet before forwarding, which adds some redundancy, on a per link basis. In case of a packet loss the receiver can recover from it during the decoding process, eliminating end-to-end multiple re-transmissions. The drawback of NC is an additional delay to encode and decode packets and a slight packet size increase. They tested NC in Mininet and concluded that it overcomes packet losses satisfactorily and, at the same time, it is also able to reduce the delay.

An et al. [75] consider a protection method that is required for potential interferences in in-band wireless SDN deployments. For this reason, they design, implement, and evaluate a proposal using ONOS and Raspberry Pi boards in an IEEE 802.11 network. They test it with a six-node mesh network, but do not provide any code repository.

Alowa et al. [76] propose an in-band control protection system based on finding a set of ideal paths for the control channel. They simulate it in diverse SNDlib networks, but do not test it with real and multiple SDN controllers.

Ochoa-Aday et al. [77] define Self-Healing Protocol (SHP) as a protocol with minimum overhead and no controller intervention. The protocol is capable of recovering in-band control plane from multiple failures in multiple-controller environments, resulting in recovery times below 20 µs, which accomplishes the 50 ms requirement of carrier-grade networks [78].

4.4. Network Bootstrapping

This section examines works that include network bootstrapping in their design. Additionally, some of them might provide automatic routing and/or failure recovery as well. As a summary, Table 4 gathers all works, ordered by publication year.

Sharma et al. [79] devise an automatic bootstrapping method to set up the in-band control connection. To the best of our knowledge, they were the first to implement an in-band configuration channel for OpenFlow. They provide a very detailed and thorough explanation of the bootstrapping process. The switch relies on Dynamic Host Configuration Protocol (DHCP) to learn the IP address and Address Resolution Protocol (ARP) to learn the MAC address of the controller. The controller uses packet probes similar to Link Layer Discovery Protocol (LLDP) to acquire the full network topology. The method is tested on emulated networks of different shapes (linear, ring, star, and mesh topologies) and sizes, using the NOX controller. The tests show minimal bootstrapping times and good scalability response.

Tu et al. [80] apply an in-band SDN control plane in which they use standard Ethernet switches on a DC network for the communication of Virtual Machines (VMs). They place a special software agent in each server and a central SDN controller with a directory server and a route algorithm server to control all communications. Based on the in-band control plane, they could also implement load balancing and fast failure recovery.

Table 4. Summary table of in-band proposals that support network boostrapping.

Article	**Year**	**Key Features**	**Evaluation & Tools**	**Available Code?**	**Overall contribution**
Sharma et al. [79]	2013	Bootstrapping and routing	Emulation: NOX/OVS/Linux namespaces/BT network, ring and mesh networks	—	★★⯪
Tu et al. [80]	2014	Bootstrapping, routing and link failure	Real testbed: Proprietary controller/1 GE and 10 GE generic switches/Spine-leaf network	—	★★⯪
Suo et al. [6]	2016	Bootstrapping, routing, link failure recovery and hybrid control mode	Emulation: Floodlight/OVS/Mininet/Fat tree network	—	★★☆
Heise et al. [81]	2017	Bootstrapping, routing and link failure	Emulation: Ryu/OVS/Mininet/linear, ring and mesh networks	—	★★☆
Su et al. [82]	2017	Bootstrapping, routing and link failure recovery	Emulation: ODL, Floodlight/OVS/Fat-tree network	—	★★⯪
Bentstuen et al. [83]	2018	Bootstrapping and routing	Emulation: ONOS/OVS/Linear network	—	★★☆
Asadujjaman et al. [84]	2018	Bootstrapping, routing and link failure recovery	Simulation: OMNeT++	—	★★☆
Lopez-Pajares et al. [85,86]	2020	Bootstrapping, routing and link failure recovery	Simulation: OMNeT++; Emulation: ONOS/BOFUSS/Waxman, Barabasi-Albert networks	✓(GitHub)	★★★
Sakic et al. [87]	2020	Bootstrapping, routing and link failure recovery	Emulation: ODL/OVS/line, star, ring, mesh networks	✓(GitHub)	★★★
Silva-Freitas et al. [88]	2020	Bootstrapping and routing	Analytical model	—	★⯪☆
Wu et al. [89]	2021	Bootstrapping	Emulation: Ryu/OVS, Mininet/Fat tree network	—	★★⯪
Li et al. [90]	2021	Bootstrapping and routing	Emulation: OVS	—	★★☆
Álvarez-Horcajo et al. [91]	2022	Bootstrapping, routing and link failure recovery	Emulation: ONOS/BOFUSS/Waxman, Barabasi-Albert networks	✓(GitHub)	★★★
Wong et al. [92]	2023	Bootstrapping, routing, link failure recovery, topology discovery and network monitoring	Emulation: BMv2/Testbed/P4Runtime	✓(GitHub)	★★★

Suo et al. [6] devise a new in-band control plane called ERIC, a hybrid-band scalable control designed for DC networks. They use pre-computed rules installed in the switches, both for the initial bootstrapping and in case of link failures or network congestion; however, they do not provide further details about it. They tested ERIC against the out-of-band approach on a Mininet simulator with the Floodlight controller and show similar RTT with a 9% throughput loss.

Heise et al. [81] propose a SDN in-band control plane to be deployed on standard avionic and industrial Ethernet networks to simplify network configuration and save additional cabling, which otherwise, means weight, complexity, and cost. Using the OpenFlow protocol in aeronautical use-cases requires: (1) the network has to be configurable via in-band, (2) the configuration must be done within a certain time, and (3) the configuration traffic must not interfere with data traffic. Two mechanisms are devised: (1) when a switch starts, it needs some preinstalled rules to connect to the controller, while at the same time, packet storm should be avoided; (2) a limitation to the in-band channel traffic based on traffic policing and admission-control techniques. They implement a proof-of-concept in Mininet and evaluate bootstrapping, fail-over times, and the influence of rate limitation in the overall configuration time under linear, ring, and mesh topologies.

Su et al. [82] define FASIC, which permits automatic bootstrapping of an OVS-based network by leveraging OVS Database (OVSDB), which allows the configuration of the SBI in the ODL controller. With the Floodlight controller, they monitor the congestion of the in-band connections. They use the ODL controller as a manager to move in-band connections from congested links, as through the OVSDB protocol, they send rules to OVS switches. They test it on a fat-tree network with ODL and Floodlight, proving recovery times that range from 0 to 5 s (faster than using stand-alone OVS, without OVSDB).

Bentstuen et al. [83] propose a technique to set up in-band control connections for dynamic environments like defense and emergency networks. They split the bootstrapping process into three steps: (1) to identify and register the new switches, (2) to set up flow rules on intermediate switches to enable a control channel, and (3) to discover and maintain the link topology database. Step 1 can be done using a registration protocol such as ARP or DHCP. In step 2, the connection is set up starting from the switch using either TCP or Transport Layer Security (TLS). The necessary flow rules on the intermediate switches must be preinstalled by the controller. Finally, for step 3, they propose to use a standard protocol like LLDP. They test their proposal using ONOS and OVS. However, they only check the case of a link failure but do not take into account the case of a node failure.

Asadujjaman et al. [84] propose a protocol for bootstrapping and routing in-band control connections, with a reactive approach for multiple link failure recovery. The only drawback is that controller-to-switch communication is source-routed, which increases packet overhead and should be improved for scalability reasons.

Lopez-Pajares et al. [85,86] present Amaru, an in-band protocol that provides bootstrapping and scalable shortest path routing for SDN networks, as well as multiple protection routes for link failures that guarantee very low fault recovery times with almost no overhead. The proposal leverages network exploration techniques to propagate a packet from the root switch (the one directly connected to the controller) to the rest of the network, in order to collect topological information, while installing the routes. In this way, each device in the network can reach the controller with the information contained in the exploration packets via various routes in order to establish the in-band control channel. However, it requires slight modifications in the SDN switches to be supported.

Sakic et al. [87] focus on the design of a full-fledged in-band control channel, considering it should implement features such as automatic bootstrapping, controller provisioning, multiple controller support, and fault resilience. It should also not be dependent on additional protocols or proprietary switch extensions. Based on these requirements, they design two approaches. The first option uses a pre-configuration to build a tree in sequential order by progressively adding new devices to the tree according to their distance from the controller. The second option relies on the Rapid Spanning Tree Protocol (RSTP) protocol to build a minimum-cost tree from the controller. Once the control channel is built using the RSTP tree, the tree is removed to avoid interference with the SDN network. Both options are very slow, providing convergence times of a few seconds, and they do not test failure recovery times.

Silva-Freitas et al. [88] propose a bootstrap protocol that provides network topology discovery and an automatic set of in-band control connections. Motivated by the high message number of OpenFlow Discovery Protocol (OFDP) and its security concerns, they propose ConForm, which is based on a spanning tree rooted at controller that spans towards every switch in the network. Initially, each switch is manually configured with a unique device ID and a cryptographic key, which only the controller can validate. When a SDN starts up, it does not forward any packet and, instead, it broadcasts a special message (reg_req) (to the controller) with its key. The controller initially only receives that message from the directly connected switches, authenticates them, and replies with their assigned ID. The process is repeated with the rest of the switches. There are additional steps to discover the topology than can be consulted in the paper. They provide an analytical model to calculate the number of exchanged messages, which is lower than other topology discovery protocols such as LLDP, OFDP, and OFDPv2. Additionally, ConForm has motivated the design of other configuration protocols, like the ETArch Switch Configuration Protocol (ETSCP) [93].

Wu et al. [89] devise a mechanism called rXstp to set up the in-band control connection with the controller. rXstp uses RSTP besides the basic layer 2 forwarding capacity of a switch to prevent loops. The node directly connected to the controller is the root node, so it is necessary to manually configure its switch priority to be the highest. When there are no more RSTP proposal messages in the network, it means that the network has converged (it is stable) and the in-band connections can be set up. The mechanism also uses a modified RSTP to allow the controller to discover the network topology. They implemented rXstp using OVS and the Ryu controller with Mininet. They compare rXstp with a SDN network that uses RSTP to set a topology and in-band control connections. They then use OFDP to discover the topology. The latter uses more messages and only discovers the tree links. This proposal only works with a single SDN controller.

Li et al. [90] propose One-Pass IBAB, a bootstrapping method for SDN in-band control, able to deploy control paths for 50 switches in 2 s. For this purpose, it builds a spanning tree rooted at the controller. If multiple controllers coexist, only the first messages exchanged are considered, so the switch connects to the first controller it is aware of. The authors envision fast failure recovery as future work.

Álvarez-Horcajo et al. [91] define an in-band control protocol with emphasis in hybrid networks (composed of both SDN and non-SDN) and also heterogeneous (both wired and wireless) devices. They evaluate their design with Waxman and Barabasi–Albert network topologies.

Wong et al. [92] propose in-band control plane called P4IBN for P4 networks with network bootstrapping, status monitoring, topology discovery, and fast failover. Among the challenges it manages to overcome is the fact that unlike OpenFlow switches, P4 switches are passive. The controller has to actively search for new P4 switches and initiate the control channel. Finally, they have been able to modify the restriction that a P4 switch can only have one control port to communicate with the controller. The evaluation was carried out using emulation and a Linux testbed. The results of the experiments indicate that the in-band control system proposed effectively carries out network bootstrapping and accomplishes rapid failover, even being able to handle multiple simultaneous failures.

4.5. Distributed Control

Finally, this section compiles surveyed works related to the use of distributed controllers (regardless of whether they provide automatic routing, failure recovery, or network bootstrapping). All articles are summarised in Table 5 and, later on, explained in detail in the rest of the section.

Table 5. Summary table of in-band proposals that support distributed control.

Article	Year	Key Features	Evaluation & Tools	Available Code?	Overall contribution
Schiff et al. [94]	2015	Bootstrapping, routing and link failure recovery, and controller-to-controller connectivity	Simulation: Java/Fat-tree network	—	★⯪☆
Schiff et al. [95]	2016	Distributed control plane that supports discovery topology and controller-to-controller connectivity	Simulation: Java	—	★★☆
Schiff et al. [96]	2016	Shared memory OF-table system with atomic operations for distributed network control	Emulation: Mininet/OVS/`ovs-ofctl` Python Wrapper	✓(GitHub)	★★☆
Görkemli et al. [97]	2016	Distributed dynamic control plane architecture	Emulation: Floodlight/OVS/Mininet	—	★⯪☆
Hark et al. [98]	2017	Control traffic isolation mechanisms and controller-to-controller connectivity	Emulation: Floodlight/OVS/Mininet	✓(GitHub)	★★☆
Canini et al. [99]	2017	Bootstrapping and topology discovery	Analytical model	—	★☆☆
Canini et al. [100]	2018	Restoration method for link-node-controller failure recovery in multi-controller networks	Emulation: Floodlight/OVS/Mininet	✓(GitHub)	★★⯪
Kwan-Yee et al. [101]	2018	Fast failure recovery mechanism in multi-controller networks	Real testbed: ONOS/HP5900	—	★⯪☆
Holzmann et al. [102]	2018	Routing for distributed controller clusters	Analytical model	—	★★☆
Canini et al. [103]	2022	Bootstrapping, routing, and link failure recovery	Emulation: Floodlight/Mininet, OVS/B4, Clos, EBONE+ networks	✓(GitHub)	★★★

Schiff et al. [94] present Medieval, which allows the switches in the network to set up an in-band connection to a single controller and also to establish routes to support connectivity among controllers. Medieval sets up the in-band control plane by creating and maintaining two spanning trees for each controller: (1) A "per-region" spanning tree spans over the region owned by a controller. The region owned by a controller is a connected graph containing the controller and the switches it controls. (2) A "network-wide" spanning tree spans over the whole network to allow controller-to-controller connectivity. However, they do not detail how the controller-to-controller connectivity is achieved. The switches need a pre-configured controller IP and a set of a priori rules for the system to work. They provide a preliminary evaluation based on a Java emulator prototype.

Schiff et al. [95] propose a plug and play distributed control plane which supports automatic topology discovery and management, as well as flexible controller membership: controllers can be added and removed dynamically. This is achieved by managing two STP trees, one for membership and is a switch with controller, and the other, which provides connectivity between controllers. The implementation relies on OpenFlow and pre-configured rules at startup, so that the network is autonomously self-stabilising. Network resilience is achieved by applying timeout policies and low-priority rules that will only take effect in the event that a switch runs out of an associated controller. Additionally, it uses ARP for topology discovery.

Schiff et al. [96] propose an in-band synchronization framework for a distributed control plane based on atomic transactions. The synchronisation primitives and atomic operations have been designed so that they can be implemented exclusively with OpenFlow. The distributed control framework could be deployed without modifying any of the code of current software switches. It makes use of OF tables to generate a shared memory system to manage the distributed control plane of the network. In order to avoid inconsistencies, a series of synchronisation mechanisms have been proposed in addition to atomic operations. The validation has been carried out using Mininet and OVS switches to test the atomic operations described above. All three previous works seem to follow a progressive evolution of their preliminary idea for distributed SDN control.

Görkemli et al. [97] propose a dynamic control plane architecture for SDN networks that can adapt to changes in high controller load and new flow demands. The proposed architecture allows the activation and deactivation of controllers on-the-fly based on automatic measurement of network control traffic and changing service requirements. The authors describe a series of base scenarios and operations that can be performed with this dynamic data plane management architecture. The proposed architecture was evaluated using Mininet, OVS switches, and Floodlight as the controller. The results show that the architecture effectively addresses bottlenecks on switch–controller links and handles high CPU usage in controllers.

Hark et al. [98] propose a low-cost inter-controller communication service, which requires few resources in terms of state, communication overhead and computational complexity, and provides isolation between control and data plane traffic using shared in-band channels. The implementation has been carried out using the Floodlight controller. It leverages port-changed event messages to embed additional information that controllers can manage to decide the best route among them. Additionally, Virtual Local Area Networks (VLANs) are employed to isolate the control traffic. They study the messages generated for the convergence of the bi-directional communication channel between controllers and its subsequent activation. They observe that the number of messages grows linearly with the number of controllers in the network.

Canini et al. [99] present two procedures to ensure that each switch in the SDN network obtains routes to support connectivity, both between every switch and all controllers, and also among controllers. These procedures are classified according to the use or lack of usage of the timeouts available in the OpenFlow tables. In the methodology that requires timeouts, low-priority rules have been used together with pre-configured anycast addresses so that the switches periodically try to connect to a new controller when they are unmanaged. The second procedure is based on running a topology discovery algorithm on each controller and iteratively adding each switch to each controller p_i on a first-come, first-served basis.

Canini et al. [100] present Renaissance as an evolution of their previously mentioned work, which is a group of algorithms for an in-band and distributed control plane for SDNs designed to overcome the shortcomings of Medieval [94]. It is not self-stabilising because its design depends on the presence of non-corrupted configuration data (e.g., the controller's IP addresses). This solution ensures that, after the occurrence of an arbitrary combination of failures, every non-faulty SDN controller can eventually reach any switch in the network within a bounded communication delay and every switch is managed by at least one non-faulty controller. The validation is accomplished using Mininet, OVS, and Floodlight. The extensive, performed tests show start-up and recovery times for different topologies (Telstra, EBONE, and Exodus). Their prototype experiments exhibit promising results. Complementarily, **M. Tran**, in his PhD thesis [104], also implements Renaissance. Preliminary tests measure bootstrapping time, number of messages, and recovery time in Mininet using OVS under DC networks. These tests show promising results as well.

Kwan-Yee et al. [101] present a software solution scheme for fast failure detection and fast failure location identification in multi-controller SDN networks. Multiple monitoring probing cycles that cover all links and nodes in the network are performed. When a failure is detected in a monitoring cycle, the controller initiates failure location identification processes to identify the failure type and the failure location. In terms of validation, they set up a real network composed of two servers running two ONOS server instances each, HP5900 switches, and two other servers as host machines. The experimental results show that the average time to recover from any single device failure is less than 50 ms, even if the failure disconnects both working control and data channels at the same time.

Holzmann et al. [102] present Seedling, an algorithm that divides the controllers managing an SDN network into groups based on their proximity. This work is the evolution of another work by the same authors, in which they presented Izzy [67], another algorithm that provided robust connectivity between controllers and all managed switches by means of shortest-path spanning trees. In this article, they propose to use Seedling to save computational cost, memory in the forwarding tables, and all the overhead that it entails at the traffic level, instead of running instances of Izzy on each controller.

Canini et al. [103] present an enhanced and comprehensively evaluated version of Renaissance [100], an in-band and distributed control plane for SDN with a particular focus on network robustness and self-stabilisation. As already known, Renaissance is based on Medieval [94] (proposed by common authors), but the former is self-stabilised, as it recovers from failures in a bounded time. Renaissance's paper provides a rigorous algorithm and an analytical proof of self-stabilisation. They comprehensively evaluate their algorithm in emulated scenarios with multiple types of networks and measuring various parameters (bootstrapping, failure recovery, etc.). Additionally, their code (based on the Floodlight controller) is publicly available.

4.6. Other Related Approaches

Although not directly implementing in-band control, **Municio** et al. [105] present Whisper, which provides end-to-end programmability for networks composed of IPv6 over the Time Synchronized Channel Hopping (TSCH) mode of IEEE 802.15.4e (6TiSCH) and wired devices. This is particularly interesting for IoT networks, in which some components might be constrained in energy or battery and out-of-band control is unfeasible. Therefore, Whisper aims to act as a middleware capable of applying directives from the SDN controllers by translating them into distributed routing messages understood by the IoT devices.

5. Discussion: Research Trends and Future Challenges

According to the performed survey, automatic routing and bootstrapping for in-band control in SDN is still an ongoing research field. Different proposals tackle both features using different techniques. Several proposals provide comprehensive implementations and evaluation, even with a publicly available code. Regarding failure recovery, some proposals exist, but just a few guarantee low recovery times with reduced overhead. From our point of view, the area could still be improved, particularly if combined with high-performance routing and automatic bootstrapping and if they are implemented in real testbeds (or at least emulated and deployed with SDN popular platforms in industry environments, like ONOS). Finally, distributed control planes still require additional research efforts. Even if several works are mentioned in this survey, most of them belong to a few authors.

Regarding research trends and future challenges, we envision potential research topics, including

- Considering IoT and Low-power and Lossy Networks (LLNs) as a key use case for leveraging in-band control, since network devices in these scenarios are usually constrained and cannot fully deploy out-of-band SDN control.
- Testing in-band approaches in big telecommunication networks (even if emulated) to check the feasibility of these approaches.
- Leveraging in-band as an East/Westbound Interface (EWBI) protocol to grant an effective and reliable communication among distributed controllers (which are the most common deployment option). This aspect is barely covered in the current literature [106].
- Evaluating potential security risks of in-band control and methods to tackle them, as just a few works mention it.
- Measuring Key Performance Indicators (KPIs) to compare in-band, out-of-band and hybrid-band approaches, as well as their implementation and compatibility with diverse SDN frameworks. Currently, few works tried to provide a holistic overview about it [7,107,108].
- Improving in-band connection routing with Artificial Intelligence (AI)/Machine Learning (ML). In computer science, AI is also known as machine intelligence. ML is a category of AI based on natural intelligence that can learn from data, make decisions, identify patterns, and perform various actions with less human intervention. ML can potentially be used to solve many problems in networking, including design, implementation, performance, and verification [109]. ML can be applied to improve in-band connectivity in many situations, such as finding the path with the highest QoS [69], finding disjoint paths for MPTCP [63,68], or improving network security by detecting a malicious node [65]. The use of ML for routing optimisation in SDN can be found in this survey [109]. A good example is the proposal by Chen [110]. They present a new routing algorithm that they implemented in a real SDN network with better results than other routing algorithms, such as OSPF and Least Loaded (LL). Ouamri et al [111] provides another example of using ML on a WAN SDN for load balancing optimisation, which could also be used for in-band backup links. ML should also be used to improve network resilience by providing some alternatives or backup paths for the in-band links.

Finally, we would like to highlight the importance of designing a full-fledged in-band protocol that covers all four aspects surveyed in this article and comprehensively integrates with an open source SDN controller like ONOS, ODL, or even TeraflowSDN [112], which was recently proposed as a SDN controller closely aligned with industry requirements and is a part of ETSI's Network Function Virtualization (NFV) and Multi-access Edge Computing (MEC) ecosystems. Providing such a holistic design would facilitate the actual usage of automatic in-band control approaches. In the meantime, while partial solutions are published, industry might still manually deploy in-band control in SDN. Nevertheless, we are aware this is an ambitious objective and would imply collaboration among more than one research group together with standardisation bodies.

6. Conclusions

In this survey, we examined all existing works in relation to in-band control in SDN. We also showed that there was some tendency in the literature for the referenced keywords SDN and in-band to grow from 2012 to 2018 and stay stable in the later years.

According to the analysis, we classified them in four main categories: (1) automatic routing: in-band control requires calculation of routes between SDN devices and the controller/s and vice versa, (2) fast failure recovery: by selecting a different route in the data plane, the control communication can be recovered after a link or switch failure, (3) network bootstrapping: refers to the ability of setting up all required parameters before starting the system, and (4) distributed control: which implies that two or more controllers are involved in the control decisions and, therefore, a coordination mechanism should be defined for routing, failure recovery, and bootstrapping.

In order to provide a comprehensive overview, we have summarised them in one table per category, including a qualitative score based on their overall contribution. These tables show the year of publication of the article, its main characteristics, the evaluation tools, whether its code is publicly available, and an overall contribution based on three stars, which was qualitatively assigned on the basis of all the previous characteristics, together with its impact in terms of number of citations.

According to our investigation, research efforts are still needed for scalable fast failure recovery and the use of in-band in distributed control scenarios. Another line of research to be promoted is the application of ML to improve in-band routing links, network security, and failure recovery.

As future research trends and challenges, we envision the importance of IoT in which some components might be constrained in energy or battery and out-of-band control is unfeasible, leveraging in-band as an EWBI protocol for controller clusters, and research on tackling potential security risks. Furthermore, most proposals cover partial aspects or are not fully integrated into real SDN platforms and devices. Therefore, there is still a need for a holistic in-band control design integrated into an industry-driven SDN framework.

Author Contributions: Conceptualization, E.R. and J.M.A.; methodology, J.M.A. and E.R.; investigation, E.R., J.M.A. and D.C.; resources, J.M.A.; papers searching and selecting, E.R. and J.M.A.; writing-original draft preparation, E.R., J.M.A. and D.C.; writing—review and editing, J.A.C., D.L.-P. and E.R.; visualization, D.L.-P. and J.A.-H.; supervision, E.R. and J.M.A.; project administration, E.R. and J.M.A.; funding acquisition, E.R. and J.A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by grants from Comunidad de Madrid through projects TAPIR-CM (S2018/TCS-4496) and MistLETOE-CM (CM/JIN/2021-006), and by project ONENESS (PID2020-116361RA-I00) of the Spanish Ministry of Science and Innovation.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

5G	The fifth generation of mobile technologies
6TiSCH	IPv6 over the TSCH mode of IEEE 802.15.4e
ARP	Address Resolution Protocol
BOFUSS	Basic OpenFlow Userspace Software Switch
BMv2	Behavioral Model version 2
DC	Data Center
DHCP	Dynamic Host Configuration Protocol
D-CPI	Data-Controller Plane Interface
EWBI	East/Westbound Interface

FRR	Fast ReRoute
HetNet	Heterogeneous Network
IoT	Internet of Things
KPI	Key Performance Indicator
LEO	Low Earth Orbit
LLDP	Link Layer Discovery Protocol
LLN	Low-power and Lossy Network
MEC	Multi-access Edge Computing
MPTCP	MultiPath TCP
NFV	Network Function Virtualization
NTN	Non-Terrestrial Network
OFDP	OpenFlow Discovery Protocol
ODL	OpenDaylight
ONF	Open Networking Foundation
ONOS	Open Network Operating System
OVS	Open vSwitch
OVSDB	OVS Database
P4	Programming Protocol-independent Packet Processors
PLR	Packet Loss Ratio
QoS	Quality of Service
RSTP	Rapid Spanning Tree Protocol
RTT	Round-Trip Time
SBI	Southbound Interface
SDN	Software-Defined Networking
SNMP	Simple Network Management Protocol
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TSCH	Time Synchronized Channel Hopping
VLAN	Virtual Local Area Network
VM	Virtual Machine

References

1. Kreutz, D.; Ramos, F.M.V.; Veríssimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
2. Silva, M.M.d.; Guerreiro, J. On the 5G and Beyond. *Appl. Sci.* **2020**, *10*, 7091. [CrossRef]
3. ONF. SDN Architecture-Issue 1.1 (ONF TR-521). 2016. Available online: https://opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf (accessed on 31 January 2023).
4. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
5. Group, T.P.A.W. P4Runtime Specification-Version 1.3.0. 2021. Available online: https://p4.org/p4-spec/p4runtime/main/P4Runtime-Spec.html (accessed on 31 January 2023).
6. Suo, C.; Tsai, I.C.; Wen, C.H.P. ERIC: Economical & reconfigurable hybrid-band control for software-defined datacenter network. In Proceedings of the 2016 International Conference on Information Networking (ICOIN), Kota Kinabalu, Malaysia, 13–15 January 2016; IEEE: New York, NY, USA, 2016; pp. 214–219. [CrossRef]
7. Jalili, A.; Nazari, H.; Namvarasl, S.; Keshtgari, M. A comprehensive analysis on control plane deployment in SDN: In-band versus out-of-band solutions. In Proceedings of the 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, Iran, 22–22 December 2017; pp. 1025–1031. [CrossRef]
8. Rojas, E. From Software-Defined to Human-Defined Networking: Challenges and Opportunities. *IEEE Netw.* **2018**, *32*, 179–185. [CrossRef]
9. Kafetzis, D.; Vassilaras, S.; Vardoulias, G.; Koutsopoulos, I. Software-Defined Networking meets Software-Defined Radio in Mobile Ad hoc Networks: State of the Art and Future Directions. *IEEE Access* **2022**, *10*, 9989–10014. [CrossRef]
10. Awan, I.I.; Shah, N.; Imran, M.; Shoaib, M.; Saeed, N. An improved mechanism for flow rule installation in-band SDN. *J. Syst. Archit.* **2019**, *96*, 1–19. [CrossRef]
11. Khorsandroo, S.; Sánchez, A.G.; Tosun, A.S.; Arco, J.; Doriguzzi-Corin, R. Hybrid SDN evolution: A comprehensive survey of the state-of-the-art. *Comput. Netw.* **2021**, *192*, 107981. [CrossRef]
12. Rojas, E.; Amin, R.; Guerrero, C.; Savi, M.; Rastegarnia, A. Challenges and Solutions for hybrid SDN. *Comput. Netw.* **2021**, *195*, 108198. [CrossRef]

13. Murtadha, M.K. SDN based device to device communication architecture for 5G mobile networks. *J. Eng. Sci. Technol.* **2021**, *16*, 3033–3047.
14. Guo, J.; Yang, L.; Liu, X.; Chen, Q.; Fan, C.; Li, X. Performance Modelling and Evaluation of In-Band Control Mode in Software-Defined Satellite Networks Based on Queuing Theory. In *Proceedings of the 2021 2nd International Conference on Computing, CNIOT '21, Networks and Internet of Things*; Association for Computing Machinery: New York, NY, USA, 2021. [CrossRef]
15. Shi, Y.; Yang, Q.; Huang, X.; Li, D.; Huang, X. An SDN-Enabled Framework for a Load-Balanced and QoS-Aware Internet of Underwater Things. *IEEE Internet Things J.* **2022**, 1. [CrossRef]
16. Maity, I.; Dhiman, R.; Misra, S. EnPlace: Energy-Aware Network Partitioning for Controller Placement in SDN. *IEEE Trans. Green Commun. Netw.* **2022**, *7*, 183–193. [CrossRef]
17. Chattopadhyay, S.; Chatterjee, S.; Nandi, S.; Chakraborty, S. Aloe: An elastic auto-scaled and self-stabilized orchestration framework for iot applications. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; IEEE: New York, NY, USA, 2019; pp. 802–810.
18. Guillen, L.; Takahira, H.; Izumi, S.; Abe, T.; Suganuma, T. On Designing a Resilient SDN C/M-Plane for Multi-Controller Failure in Disaster Situations. *IEEE Access* **2020**, *8*, 141719–141732. [CrossRef]
19. Guillen, L.; Izumi, S.; Abe, T.; Suganuma, T. A Resilient Mechanism for Multi-Controller Failure in Hybrid SDN-based Networks. In Proceedings of the 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), Tainan, Taiwan, 8–10 September 2021; IEEE: New York, NY, USA, 2021; pp. 285–290.
20. Neves, R.H.; Silva, A.A.; Gava, V.; Azevedo, M.T.; Sandoval, J.F.; Oliveira, F.S.; Guelfi, A.E.; Kofuji, S.T. DoS Attack on SDN: A study on control plane strategies in-band and out-of-band. *Res. Sq.* **2022**. [CrossRef]
21. Nunes, B.A.A.; Mendonca, M.; Nguyen, X.N.; Obraczka, K.; Turletti, T. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1617–1634. [CrossRef]
22. Jarraya, Y.; Madi, T.; Debbabi, M. A Survey and a Layered Taxonomy of Software-Defined Networking. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1955–1980. [CrossRef]
23. Rowshanrad, S.; Namvarasl, S.; Abdi, V.; Hajizadeh, M.; Keshtgary, M. A survey on SDN, the future of networking. *J. Adv. Comput. Sci. Technol.* **2014**, *3*, 232–248. [CrossRef]
24. Jammal, M.; Singh, T.; Shami, A.; Asal, R.; Li, Y. Software defined networking: State of the art and research challenges. *Comput. Netw.* **2014**, *72*, 74–98. [CrossRef]
25. Farhady, H.; Lee, H.; Nakao, A. Software-Defined Networking: A survey. *Comput. Netw.* **2015**, *81*, 79–95. [CrossRef]
26. Xia, W.; Wen, Y.; Foh, C.H.; Niyato, D.; Xie, H. A Survey on Software-Defined Networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 27–51. [CrossRef]
27. Benzekki, K.; El Fergougui, A.; Elbelrhiti Elalaoui, A. Software-defined networking (SDN): A survey. *Secur. Commun. Netw.* **2016**, *9*, 5803–5833. Available online: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.1737 (accessed on 31 January 2023). [CrossRef]
28. Masoudi, R.; Ghaffari, A. Software defined networks: A survey. *J. Netw. Comput. Appl.* **2016**, *67*, 1–25. [CrossRef]
29. Singh, S.; Jha, R.K. A survey on software defined networking: Architecture for next generation network. *J. Netw. Syst. Manag.* **2017**, *25*, 321–374. [CrossRef]
30. Scott-Hayward, S.; O'Callaghan, G.; Sezer, S. Sdn Security: A Survey. In Proceedings of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS), Trento, Italy, 11–13 November 2013; pp. 1–7. [CrossRef]
31. Ahmad, I.; Namal, S.; Ylianttila, M.; Gurtov, A. Security in Software Defined Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2317–2346. [CrossRef]
32. Ali, S.T.; Sivaraman, V.; Radford, A.; Jha, S. A Survey of Securing Networks Using Software Defined Networking. *IEEE Trans. Reliab.* **2015**, *64*, 1086–1097. [CrossRef]
33. Correa Chica, J.C.; Imbachi, J.C.; Botero Vega, J.F. Security in SDN: A comprehensive survey. *J. Netw. Comput. Appl.* **2020**, *159*, 102595. [CrossRef]
34. Amin, R.; Reisslein, M.; Shah, N. Hybrid SDN Networks: A Survey of Existing Approaches. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3259–3306. [CrossRef]
35. Sandhya; Sinha, Y.; Haribabu, K. A survey: Hybrid SDN. *J. Netw. Comput. Appl.* **2017**, *100*, 35–55. [CrossRef]
36. Oktian, Y.E.; Lee, S.; Lee, H.; Lam, J. Distributed SDN controller system: A survey on design choice. *Comput. Netw.* **2017**, *121*, 100–111. [CrossRef]
37. Bannour, F.; Souihi, S.; Mellouk, A. Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 333–354. [CrossRef]
38. Bizanis, N.; Kuipers, F.A. SDN and Virtualization Solutions for the Internet of Things: A Survey. *IEEE Access* **2016**, *4*, 5591–5606. [CrossRef]
39. Bera, S.; Misra, S.; Vasilakos, A.V. Software-Defined Networking for Internet of Things: A Survey. *IEEE Internet Things J.* **2017**, *4*, 1994–2008. [CrossRef]
40. Trois, C.; Del Fabro, M.D.; de Bona, L.C.E.; Martinello, M. A Survey on SDN Programming Languages: Toward a Taxonomy. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2687–2712. [CrossRef]
41. Ali, J.; Lee, G.M.; Roh, B.H.; Ryu, D.K.; Park, G. Software-Defined Networking Approaches for Link Failure Recovery: A Survey. *Sustainability* **2020**, *12*, 4255. [CrossRef]

42. Karakus, M.; Durresi, A. Quality of Service (QoS) in Software Defined Networking (SDN): A survey. *J. Netw. Comput. Appl.* **2017**, *80*, 200–218. [CrossRef]
43. Tan, L.; Su, W.; Zhang, W.; Lv, J.; Zhang, Z.; Miao, J.; Liu, X.; Li, N. In-band Network Telemetry: A Survey. *Comput. Netw.* **2021**, *186*, 107763. [CrossRef]
44. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. In-band control, queuing, and failure recovery functionalities for openflow. *IEEE Netw.* **2016**, *30*, 106–112. [CrossRef]
45. Bosshart, P.; Daly, D.; Gibb, G.; Izzard, M.; McKeown, N.; Rexford, J.; Schlesinger, C.; Talayco, D.; Vahdat, A.; Varghese, G.; et al. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–95. [CrossRef]
46. Hauser, F.; Häberle, M.; Merling, D.; Lindner, S.; Gurevich, V.; Zeiger, F.; Frank, R.; Menth, M. A survey on data plane programming with P4: Fundamentals, advances, and applied research. *J. Netw. Comput. Appl.* **2023**, *212*, 103561. [CrossRef]
47. Berde, P.; Gerola, M.; Hart, J.; Higuchi, Y.; Kobayashi, M.; Koide, T.; Lantz, B.; O'Connor, B.; Radoslavov, P.; Snow, W.; et al. ONOS: Towards an Open, Distributed SDN OS. In Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN '14, Chicago, IL, USA, 22 August 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 1–6. [CrossRef]
48. Medved, J.; Varga, R.; Tkacik, A.; Gray, K. OpenDaylight: Towards a Model-Driven SDN Controller architecture. In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, Sydney, Australia, 19 June 2014; pp. 1–6. [CrossRef]
49. Tomonori, F. Introduction to Ryu SDN framework. *Open Netw. Summit* **2013**, 1–14.
50. Pfaff, B.; Pettit, J.; Koponen, T.; Jackson, E.; Zhou, A.; Rajahalme, J.; Gross, J.; Wang, A.; Stringer, J.; Shelar, P.; et al. The Design and Implementation of Open vSwitch. In Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), Oakland, CA, USA, 4–6 May 2015; USENIX Association: Oakland, CA, USA, 2015; pp. 117–130.
51. Fernandes, E.L.; Rojas, E.; Alvarez-Horcajo, J.; Kis, Z.L.; Sanvito, D.; Bonelli, N.; Cascone, C.; Rothenberg, C.E. The road to BOFUSS: The basic OpenFlow userspace software switch. *J. Netw. Comput. Appl.* **2020**, *165*, 102685. [CrossRef]
52. P4. Behavioral Model (bmv2): The Reference P4 Software Switch. Available online: https://github.com/p4lang/behavioral-model (accessed on 31 January 2023).
53. Lantz, B.; Heller, B.; McKeown, N. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets-IX, Monterey, CA, USA, 20–21 October 2010; Association for Computing Machinery: New York, NY, USA, 2010. [CrossRef]
54. Varga, A.; Hornig, R. An Overview of the OMNeT++ Simulation Environment. In Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems, Workshops, Simutools '08, Marseille, France, 3–7 March 2008; ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Brussels, Belgium, 2008. [CrossRef]
55. Varga, A.OMNeT++. In *Modeling and Tools for Network Simulation*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 35–59. [CrossRef]
56. Henderson, T.R.; Lacage, M.; Riley, G.F.; Dowell, C.; Kopena, J. Network simulations with the ns-3 simulator. *SIGCOMM Demonstr.* **2008**, *14*, 527.
57. Riley, G.F.; Henderson, T.R. The ns-3 Network Simulator. In *Modeling and Tools for Network Simulation*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 15–34. [CrossRef]
58. Rojas, E.; Doriguzzi-Corin, R.; Tamurejo, S.; Beato, A.; Schwabe, A.; Phemius, K.; Guerrero, C. Are We Ready to Drive Software-Defined Networks? A Comprehensive Survey on Management Tools and Techniques. *ACM Comput. Surv.* **2018**, *51*, 1–35. [CrossRef]
59. Waxman, B.M. Routing of multipoint connections. *IEEE J. Sel. Areas Commun.* **1988**, *6*, 1617–1622. [CrossRef]
60. Barabási, A.L.; Albert, R. Emergence of scaling in random networks. *Science* **1999**, *286*, 509–512. [CrossRef]
61. Goltsmann, P.; Zitterbart, M.; Hecker, A.; Bless, R. Towards a Resilient In-Band SDN Control Channel. *Universität Tübingen* **2017**. [CrossRef]
62. Khakhalin, A.S.; Chemeritskiy, E.V. A reliable in-band control in a Software-Defined Network. *J. Theor. Appl. Inf. Technol.* **2017**, *95*, 4283–4290.
63. Raza, A.; Gohar, A.; Lee, S. MPTCP based in-band controlling for the software defined networks. In Proceedings of the 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, 18–20 October 2017; pp. 163–167. [CrossRef]
64. González, S.; De la Oliva, A.; Bernardos, C.J.; Contreras, L.M. Towards a Resilient Openflow Channel Through MPTCP. In Proceedings of the 2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Valencia, Spain, 6–8 June 2018; pp. 1–5. [CrossRef]
65. Mohan, P.M.; Truong-Huu, T.; Gurusamy, M. Towards resilient in-band control path routing with malicious switch detection in SDN. InProceedings of the 2018 10th International Conference on Communication Systems Networks (COMSNETS), Bengaluru, India, 3–7 January 2018; pp. 9–16. [CrossRef]
66. Görkemli, B.; Tatlıcıoğlu, S.; Tekalp, A.M.; Civanlar, S.; Lokman, E. Dynamic Control Plane for SDN at Scale. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2688–2701. [CrossRef]

67. Holzmann, P.; Zitterbart, M. Izzy: A Distributed Routing Protocol for In-band SDN Control Channel Connectivity. In Proceedings of the 2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium), Osnabrück, Germany, 14–17 October 2019; pp. 18–25. Available online: https://doi.org/jt8q (accessed on 31 January 2023). [CrossRef]
68. Raza, A.; Lee, S. Gate Switch Selection for In-Band Controlling in Software Defined Networking. *IEEE Access* **2019**, *7*, 5671–5681. [CrossRef]
69. Fan, W.; Yang, F. Centralized Trust-Based In-Band Control for SDN Control Channel. *IEEE Access* **2020**, *8*, 4289–4300. [CrossRef]
70. Ningyuan, W.; Chen, D.; Liang, L.; Wang, M.; Bingyuan, L. An SDN Based Highly Reliable in-Band Control Framework for LEO Mega-Constellations. In Proceedings of the 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), Guangzhou, China, 23–26 April 2021; pp. 970–975. [CrossRef]
71. Kumazoe, K.; Shibata, M.; Tsuru, M. A P4 BMv2-Based Feasibility Study on a Dynamic In-Band Control Channel for SDN. In *Proceedings of the Advances in Intelligent Networking and Collaborative Systems*; Barolli, L., Miwa, H., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 442–451.
72. Huang, H.; Guo, S.; Liang, W.; Li, K.; Ye, B.; Zhuang, W. Near-Optimal Routing Protection for In-Band Software-Defined Heterogeneous Networks. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 2918–2934. [CrossRef]
73. Park, Y.; Nguyen, D.T.; Kang, B.; Lee, K.; Lee, J.; Choo, H. A Fast Recovery Scheme Based on Detour Planning for In-Band Openflow Networks. InProceedings of the 11th International Conference on Ubiquitous Information Management and Communication, IMCOM '17, Beppu, Japan, 5–7 January 2017; Association for Computing Machinery: New York, NY, USA, 2017. [CrossRef]
74. Akhtar, F.; Rehmani, M.H.; Davy, A. A Network Coding Approach to In-Band Control Traffic Sharing in Software Defined Networks. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 267–271. [CrossRef]
75. An, N.; Lim, H. Poster: Protecting Control Planes in In-Band Software-Defined Wireless Networks. In Proceedings of the The 25th Annual International Conference on Mobile Computing and Networking, MobiCom '19, Los Cabos, Mexico, 21–25 October 2019; Association for Computing Machinery: New York, NY, USA, 2019. [CrossRef]
76. Alowa, A.; Fevens, T. A Dynamic Recovery Module for In-band Control Channel Failure In Software Defined Networking. In Proceedings of the 2020 6th IEEE Conference on Network Softwarization (NetSoft), Virtual Conference, 29 June–3 July 2020; pp. 209–217. [CrossRef]
77. Ochoa-Aday, L.; Cervelló-Pastor, C.; Fernández-Fernández, A. Self-healing and SDN: Bridging the gap. *Digit. Commun. Netw.* **2020**, *6*, 354–368. [CrossRef]
78. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. OpenFlow: Meeting Carrier-Grade Recovery Requirements. *Comput. Commun.* **2013**, *36*, 656–665. [CrossRef]
79. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. Automatic bootstrapping of OpenFlow networks. In Proceedings of the 2013 19th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN), Brussels, Belgium, 10–12 April 2013; pp. 1–6. [CrossRef]
80. Tu, C.C.; Wang, P.W.; Chiueh, T.c. In-Band Control for an Ethernet-Based Software-Defined Network. In Proceedings of the International Conference on Systems and Storage, SYSTOR 2014, Haifa, Israel, 10–12 June 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 1–11. [CrossRef]
81. Heise, P.; Geyer, F.; Obermaisser, R. Self-configuring deterministic network with in-band configuration channel. In Proceedings of the 2017 Fourth International Conference on Software Defined Systems (SDS), Valencia, Spain, 8–11 May 2017; pp. 162–167. [CrossRef]
82. Su, Y.L.; Wang, I.C.; Hsu, Y.T.; Wen, C.H.P. FASIC: A Fast-Recovery, Adaptively Spanning In-Band Control Plane in Software-Defined Network. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6. [CrossRef]
83. Bentstuen, O.I.; Flathagen, J. On Bootstrapping In-Band Control Channels in Software Defined Networks. In Proceedings of the 2018 IEEE International Conference on Communications Workshops (ICC Workshops), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6. [CrossRef]
84. Asadujjaman, A.S.M.; Rojas, E.; Alam, M.S.; Majumdar, S. Fast Control Channel Recovery for Resilient In-band OpenFlow Networks. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 19–27. [CrossRef]
85. Lopez-Pajares, D.; Alvarez-Horcajo, J.; Rojas, E.; Asadujjaman, A.S.M.; Martinez-Yelmo, I. Amaru: Plug&Play Resilient In-Band Control for SDN. *IEEE Access* **2019**, *7*, 123202–123218. [CrossRef]
86. Constantin, B.N. Desarrollo de una solución de encaminamiento para tráfico de control in-band en entornos SDN. Master's Thesis, Universidad de Alcalá, Escuela Politécnica Superior, Alcalá de Henares, Spain, 2020; pp. 1–162.
87. Sakic, E.; Avdic, M.; Van Bemten, A.; Kellerer, W. Automated Bootstrapping of A Fault-Resilient In-Band Control Plane. In Proceedings of the Symposium on SDN Research, SOSR '20, San Jose, CA, USA, 3 March 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1–13. [CrossRef]
88. Silva Freitas, M.; Oliveira, R.; Molinos, D.; Melo, J.; Frosi Rosa, P.; de Oliveira Silva, F. ConForm: In-band Control Plane Formation Protocol to SDN-Based Networks. In Proceedings of the 2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 7–10 January 2020; pp. 574–579. [CrossRef]

89. Wu, F.; Tian, A. rXstp: A Topology Discovery Mechanism Based on Rapid Spanning Tree for SDN In-Band Control. In Proceedings of the 2021 International Conference on Communications, Information System and Computer Engineering (CISCE), Beijing, China, 14–16 May 2021; pp. 703–706. [CrossRef]
90. Li, C.Y.; Yen, L.H.; Chi, K.H.; Tseng, C.C. One-Pass In-Band Automatic Bootstrapping for OpenFlow Switches. *IEEE Access* **2021**, *9*, 153349–153359. [CrossRef]
91. Alvarez-Horcajo, J.; Martinez-Yelmo, I.; Rojas, E.; Carral, J.A.; Carrascal, D. ieHDDP: An Integrated Solution for Topology Discovery and Automatic In-Band Control Channel Establishment for Hybrid SDN Environments. *Symmetry* **2022**, *14*, 756. [CrossRef]
92. Wong, T.S.; Lee, S.S.W. Design of an In-Band Control Plane for Automatic Bootstrapping and Fast Failure Recovery in P4 Networks. *IEEE Trans. Netw. Serv. Manag.* 2023, *in press*. [CrossRef]
93. Oliveira, R.D.; Freitas, M.S.; Molinos, D.N.; Rosa, P.F.; Mesquita, D.G. ETSCP: Flexible SDN data plane configuration based on bootstrapping of in-band control channels. In Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 17–21 May 2021; pp. 711–715.
94. Schiff, L.; Schmid, S.; Canini, M. Medieval: Towards A Self-Stabilizing, Plug & Play, In-Band SDN Control Network. In Proceedings of the ACM Sigcomm Symposium on SDN Research (SOSR), Santa Clara, CA, USA, 17–18 June 2015.
95. Schiff, L.; Schmid, S.; Canini, M. Ground control to major faults: Towards a fault tolerant and adaptive SDN control network. In Proceedings of the 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), Toulouse, France, 28 June–1 July 2016; IEEE: New York, NY, USA, 2016; pp. 90–96. [CrossRef]
96. Schiff, L.; Schmid, S.; Kuznetsov, P. In-Band Synchronization for Distributed SDN Control Planes. *SIGCOMM Comput. Commun. Rev.* **2016**, *46*, 37–43. [CrossRef]
97. Görkemli, B.; Parlakışık, A.M.; Civanlar, S.; Ulaş, A.; Tekalp, A.M. Dynamic management of control plane performance in software-defined networks. In Proceedings of the 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, Korea, 6–10 June 2016; IEEE: New York, NY, USA, 2016; pp. 68–72.
98. Hark, R.; Rizk, A.; Richerzhagen, N.; Richerzhagen, B.; Steinmetz, R. Isolated in-band communication for distributed SDN controllers. In Proceedings of the 2017 IFIP Networking Conference (IFIP Networking) and Workshops, Stockholm, Sweden, 12–15 June 2017; pp. 1–2. Available online:https://doi.org/jvb5 (accessed on 31 January 2023). [CrossRef]
99. Canini, M.; Salem, I.; Schiff, L.; Schiller, E.M.; Schmid, S. A Self-Organizing Distributed and In-Band SDN Control Plane. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 2656–2657. [CrossRef]
100. Canini, M.; Salem, I.; Schiff, L.; Schiller, E.M.; Schmid, S. Renaissance: A Self-Stabilizing Distributed SDN Control Plane. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–6 July 2018; pp. 233–243. [CrossRef]
101. Chan, K.Y.; Chen, C.H.; Chen, Y.H.; Tsai, Y.J.; Lee, S.S.W.; Wu, C.S. Fast Failure Recovery for In-Band Controlled Multi-Controller OpenFlow Networks. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 17–19 October 2018; pp. 396–401. [CrossRef]
102. Holzmann, P.; Hecker, A.; Zitterbart, M. Towards a Distributed Routing Protocol for In-Band Control Channel with Elastic Controller Clusters. InProceedings of the 2018 IEEE Globecom Workshops (GC Wkshps), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6. [CrossRef]
103. Canini, M.; Salem, I.; Schiff, L.; Schiller, E.M.; Schmid, S. Renaissance: A self-stabilizing distributed SDN control plane using in-band communications. *J. Comput. Syst. Sci.* **2022**, *127*, 91–121. [CrossRef]
104. Tran, M. A Floodlight Extension for Supporting a Self-Stabilizing In-Band Control Plane for Software Defined Networking. Master's Thesis, Computer Science and Engineering, Chalmers Lindholmen, Gothenburg, Sweden, 2018.
105. Municio, E.; Balemans, N.; Latré, S.; Marquez-Barja, J. Leveraging Distributed Protocols for full End-to-End Softwarization in IoT Networks. In Proceedings of the 2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2020; pp. 1–6. [CrossRef]
106. Saeed, K.; Ullah, M.O. Toward Reliable Controller Placements in Software-Defined Network Using Constrained Multi-Objective Optimization Technique. *IEEE Access* **2022**, *10*, 129865–129883. [CrossRef]
107. Wazirali, R.; Ahmad, R.; Alhiyari, S. SDN-openflow topology discovery: An overview of performance issues. *Appl. Sci.* **2021**, *11*, 6999. [CrossRef]
108. Zopellaro Soares, A.A.; Lucas Vieira, J.; Quincozes, S.E.; Ferreira, V.C.; Uchôa, L.M.; Lopes, Y.; Passos, D.; Fernandes, N.C.; Monteiro Moraes, I.; Muchaluat-Saade, D.; et al. SDN-based teleprotection and control power systems: A study of available controllers and their suitability. *Int. J. Netw. Manag.* **2021**, *31*, e2112. [CrossRef]
109. Amin, R.; Rojas, E.; Aqdus, A.; Ramzan, S.; Casillas-Perez, D.; Arco, J.M. A Survey on Machine Learning Techniques for Routing Optimization in SDN. *IEEE Access* **2021**, *9*, 104582–104611. [CrossRef]
110. Chen, Y.R.; Rezapour, A.; Tzeng, W.G.; Tsai, S.C. RL-routing: An SDN routing algorithm based on deep reinforcement learning. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 3185–3199. [CrossRef]

111. Ouamri, M.A.; Barb, G.; Singh, D.; Alexa, F. Load Balancing Optimization in Software-Defined Wide Area Networking (SD-WAN) using Deep Reinforcement Learning. In Proceedings of the 2022 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 10–11 November 2022; IEEE: New York, NY, USA, 2022; pp. 1–6.
112. Vilalta, R.; Muñoz, R.; Casellas, R.; Martínez, R.; López, V.; de Dios, O.G.; Pastor, A.; Katsikas, G.P.; Klaedtke, F.; Monti, P.; et al. TeraFlow: Secured Autonomic Traffic Management for a Tera of SDN flows. In Proceedings of the 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Porto, Portugal, 8–11 June 2021; pp. 377–382. [CrossRef]