Journal of
Sensor and Actuator Networks

MDPI

Article

Hybrid SDN Performance: Switching between Centralized and Distributed Modes under Unreliable Control Communication Channels †

Mohammed Osman [1,*] and Josep Mangues-Bafalluy [2]

[1] Departamento de Ingeniería Telemática, Universitat Politècnica de Catalunya (UPC), 08034 Barcelona, Spain
[2] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), 08860 Castelldefels, Spain; josep.mangues@cttc.cat
* Correspondence: mohammed.osman.osman@upc.edu
† This paper is an extended version of our paper published in Osman, M.; Núñez-Martínez, J.; Mangues-Bafalluy, J. Hybrid sdn: Evaluation of the impact of an unreliable control channel. In Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks, Berlin, Germany, 6–8 November 2017; pp. 242–246.

Abstract: Software-defined networking generally assumes ideal control channels between controller and network nodes. This may not be the case in challenged environments that are becoming more common due to dense and reduced-coverage 5G deployments and use cases requiring cost-effective wireless transport networks. In this paper, we evaluate the impact on network performance of unreliable controller-to-node communication channels, propose a hybrid SDN (hSDN) solution that switches between centralized and distributed operational modes depending on network conditions, and evaluate this solution under a variety of network scenarios (e.g., link impairments or packet loss ratios) designed to assess its operational limits. The results show that the proposed solution substantially improved the aggregated throughput, particularly when control channel packet loss ratios increased, while only showing a slight increase in average latency (e.g., 28% throughput improvement for 20% control packet losses). This enables network operation in hard conditions under which a canonical centralized SDN control would result in a nonoperational network.

Keywords: SDN; hybrid SDN; wireless transport networks; reliability; centralized-distributed control

Citation: Osman, M.; Mangues-Bafalluy, J. Hybrid SDN Performance: Switching between Centralized and Distributed Modes under Unreliable Control Communication Channels. *J. Sens. Actuator Netw.* **2021**, *10*, 57. https://doi.org/10.3390/jsan10030057

Academic Editors: Ricardo Severino and Hossein Fotouhi

Received: 2 June 2021
Accepted: 17 August 2021
Published: 20 August 2021

1. Introduction

Adding to the current trend of a steep mobile data increase due to massive Machine Type Communication (mMTC), novel 5G network use cases are only expected to contribute even more to this trend. For monitoring and sensing of mMTC, Wireless Sensor Actuator Networking (WSAN) technologies play an important role. The incorporation of 5G technology on the underlying WSAN may meet the throughput demand of anticipated data growth. To overcome the limited availability of the radio spectrum in current mobile networks (e.g., Long Term Evolution or LTE), the move toward higher frequencies in search of larger spectrum chunks comes naturally with small cell deployments.

By deploying small cells (SC) [1,2] with shorter cell radii, the capacity of the mobile networks can be increased by spatial reuse of the radio spectrum. The massive deployment of SCs, in turn, also comes with a variety of transport network options, including wireless transport (e.g., when deployed in lampposts). Multi-hop wireless networks (e.g., microwave or mm-Wave backhaul links) to interconnect the SCs could be a good choice to provide a cost-effective mobile transport solution (including backhaul and fronthaul). In fact, wireless transport is expected to reach a 47% share in 2026 as small cell backhaul [3] due to unavailability and/or the cost of wired transport.

On the network management/control front, network programmability and virtualization are also an integral part of 5G networks. Management and orchestration frameworks

are being designed by various groups/organizations (e.g., ETSI Network Function Virtualization), which aim at an end-to-end orchestration of all network segments from RAN to core all the way through transport. On the transport side (the focus of this paper), Software-Defined Networking (SDN) has also been integrated in these frameworks [4] as a fundamental network programmability enabler.

The SDN paradigm separates the data and control planes of the network. It logically centralizes the control of a network in an SDN controller which acts as a brain of the network and is in charge of telling each network node how to forward incoming packets by installing the appropriate forwarding rules. One of the main advantages it brings is programmability through this single entity (the logical controller) with which network management applications must interact to apply their policies. Through agreed-upon APIs, the full potential of SDN can be exploited by the network managers.

This has been the case in controlled scenarios that do not pose challenging constraints to the deployment of such a paradigm because of the availability of reliable and high capacity networks, for instance, in data centers. However, moving to wide area networks (WAN) (e.g., microwave or mm-Wave backhaul links) where in-band channels are responsible to carry control traffic between the nodes and the SDN controller, the assumption of the availability of such a reliable network may not hold anymore as performance of the wireless link changes with the environmental conditions, which leads to a high risk of experiencing channel impairments, which might cause centralized SDN operation failure by affecting communication between the transport component of SCs and the SDN controller.

Therefore, although SDN provides huge advantages for managing networks by splitting the control logic, there are some pitfalls of such separation in the presence of unreliable conditions, namely (i) *data plane faults* (the network element(s) or port(s) associated to the network element(s) fails), (ii) *control channel faults* (the connection between the SDN controller and the data plane element(s) fails or it experiences losses), and (iii) *SDN controller faults* (the SDN controller fails).

For instance, when using OpenFlow [5], control messages that are fundamental for installing packet forwarding rules (e.g., PacketIn and PacketOut) and node handling (e.g., PortStatus) or topology discovery messages (Link Layer Discovery Protocol, or LLDP) may be lost. TCP retransmissions may be able to handle some low loss situations; however, a more detailed analysis is needed. Therefore, handling such impairments is key for the correct operation of the transport network in such environments.

Previous work explored the application of SDN to wireless transport networks [6–8], though it is generally assumed that unreliability and failures may only happen in the data plane. In fact, previous studies mainly focused on data plane reliability by designing efficient schemes for fast detection and recovery of the communication [9–11]. Controller-only schemes have also been integrated, for instance in ONOS, in which complex cluster management and device mastership procedures were put in place toward consistency in the case of the controller failure [12]. Additionally, hSDN approaches have also appeared for allowing the transition of networks from distributed to SDN-based operation [13], which also offers interesting ideas that can be applied to our problem.

This work presents a hSDN-based reliable control plane that reacts on control communication impairments, and even, controller failure. Our scheme differs from state-of-the-art solutions in the way the control plane failure is handled. We considered a metric named Control Packet Loss Ratio (CPLR). As control packet loss is our major concern, from the point of view of a network node, the failure of the SDN controller is equivalent to the failure of the control communication channel.

Thus, we propose to look at control plane reliability from the point of view of the node that experiences the impairments. We introduce a local agent in the nodes to detect unreliability of the control plane communication channel and a control logic switching algorithm to make a decision for whether to operate in a centralized or distributed way. Therefore, our scheme provides a solution not only for the control communication channel failure but also for the SDN controller failure as well.

In this sense, this article presents a scheme that explores the benefits of centralized and distributed operations depending on control communication channel conditions. It combines both modes of operation into the same node; hence, the hSDN approach is generalized in the network and not just used in selected nodes. Some preliminary evaluations were presented in ref. [14], which indicated the impact that an unreliable control plane may have over the data plane. In this sense, it analyzed the dimension of the problem.

Based on these initial findings, this paper further refines the preliminary thoughts mentioned in that paper toward more elaborate and automated decision making by presenting a complete solution and exhaustively evaluating it for various network sizes and diverse network impairments. We also emulate different network topologies under various traffic conditions. This article compares centralized and distributed operations under unreliable conditions to find the correct network operation switching point between centralized and distributed operations.

In this direction, we deploy a hSDN scheme that changes the mode of operation at the node level when needed, based on the conceived control logic switching algorithm. The results show that, for the evaluated scenarios, the proposed hSDN approach maintained the network operational by achieving the best performance of both modes under all conditions. In the case of high loss regimes, the proposed hSDN scheme improved the throughput and maintained acceptable latency compared to the purely distributed approach.

The rest of sections of the paper are organized as follows. Section 2 presents the current state of the art. Section 3 describes the proposed DenseNet-hybrid control plane architecture, including proposed changes in data plane node architecture with an integrated local agent. The control logic switching algorithm is also described in this section. The proposed approach is evaluated in Section 4. Section 5 presents a comparison of the proposed approach to the existing approaches, and Section 6 discusses the performance metrics. Finally, our conclusions are presented in Section 7.

2. Related Work

The separation of the control plane from the data plane in SDN architecture introduces an extra point of failure which is the control plane failure. Little effort has been devoted by the community to solve the shortcomings of the SDN paradigm under control plane unreliability (including controller failures and control channel losses).

In order to overcome SDN controller failure, the deployment of multiple SDN controllers [5,12] may be a solution (controller redundancy); however, it may also lead to inconsistency issues. For instance, HyperFlow [15] proposes physically distributed controllers that are synchronized through a publish/subscribe system. ONOS requires complex procedures for inter-controller synchronization. Another option is to design schemes to handle control plane unreliability.

For instance, ResilientFlow [16] restores the control channel through alternate paths (path redundancy) in the presence of channel failure. However, these research works focused on solving a single specific type of faults, such as SDN controller failure or control channel failure. Moreover, to tackle SDN controller failures, a solution may be to form a hierarchy or cascade deployment model. In this context, OpenFlow, since version 1.3 [5], supports multiple SDN controllers for managing an equivalent set of forwarding nodes. However, some inconsistent issues, like Event Ordering, Unreliable Event Delivery, and Repletion of Commands, are still matters of concern.

Ravana [17] ensured event ordering, correct event processing, and execution of commands for exactly once during SDN controller failure. In ref. [18], an integrated SDN concept was proposed where an OLSR-to-OpenFlow (O2O) module was presented to configure control rules that are used to forward OpenFlow packets. After detection of SDN controller failure by the O2O module, the O2O module dumps the OLSR routing tables into the forwarding nodes to recover from SDN controller failure. This approach needs the translation of routing tables into SDN rules.

Although there have been complex schemes applied in SDN controllers to handle cluster management, device mastership to guarantee consistency in case of controller failure [12], to our knowledge, limited efforts have been devoted to handle degraded control channel performance (e.g., a lossy control channel), which may be the norm when deploying multi-hop wireless networks to serve dense small cell deployments, for instance. Contrarily, this paper focuses precisely on this by proposing a simple (hence, fast to execute/decide) scheme that recovers from SDN controller fault, control channel failure, or even degraded control channel by reacting to Control Packet Loss Ratio (CPLR) variations of the control communication channels as a measure of the unreliability of the control plane from the viewpoint of the nodes.

3. DenseNet-Hybrid Control Plane Architecture

In this section, we first illustrate our proposed architecture to form the DenseNet-hybrid SDN infrastructure that adapts the wireless mesh transport network composed of transport nodes co-located with SCs to provide connectivity between these SCs. Second, we describe a hybrid node architecture where both centralized and distributed operations coexist and an integrated local agent that is based on a monitoring framework, periodically monitors control plane for the occurrence of any impairments. Finally, we describe the network logic switching algorithm that is integrated into the nodes to perform network operation switching between centralized and distributed modes in our hybrid control plane architecture.

3.1. DenseNet-Hybrid SDN Infrastructure

The idea of deploying SCs is to enhance the coverage area and capacity offered by macro-cells by adding more base stations with smaller coverage. The SCs are responsible to connect the sensor devices as well as other Internet of Things (IoT) devices to the mobile network. The macro-cell site usually acts as a gateway site for aggregating the traffics of a set of SCs and provides connectivity to the core network.

Due to the capacity concern of mobile networks, fiber-based technology can be a good option for the mobile backhaul. However, due to implementation cost (e.g., laying fiber to all the SCs) and the nature of the area to cover, it may not be very cost-effective. On the other hand, a multi-hop wireless backhaul, by interconnecting SCs with wireless links (e.g., microwave or mm-Wave backhaul links), can provide a cost-effective solution. The deployment of SCs in mobile backhaul networks can greatly enhance network capacity by reusing the radio spectrum [1].

In this context, by maintaining the global view of the network, SDN can provide huge advantages for managing the network and to provide better resource allocation. Even though SDN provides better manageability of the network, in wireless networks, where control messages are sent over in-band channels, the performance of the SDN-based network may degrade due to channel impairments (e.g., the loss of control messages). In general, in mmWave mobile backhaul, mesh topology is an ideal candidate to interconnect the SCs where SC base station is placed on the street furniture (e.g., a lamp post). In such a scenario, due to obstructions, all the SCs may not be connected directly to the aggregation point (e.g., macro-cell).

Only one SC may be connected directly to the aggregation point and act as a gateway node for other SCs that are in the coverage area of the macro-cell. The SDN controller may be placed in the aggregation point (see Figure 1). Communication between the SDN controller and the SCs will be maintained via gateway SC, which is a realistic in-band SDN deployment.

As Figure 1 illustrates, we propose a hybrid control plane architecture for DenseNet. In this way, our architecture attempts to preserve the benefits of both worlds (i.e., centralized and distributed). Specifically, we propose to maintain a centralized control logic to preserve the benefits of canonical-SDN (i.e., simple network management, programmability) under reliable control plane conditions, whereas the distributed control plane is in

charge of acting under unreliable conditions to quickly react to failures and to avoid the inefficient use of an inaccessible (or hard to reliably reach) centralized control logic. In this way, network nodes are modified to be able to dynamically switch among both modes of operation.

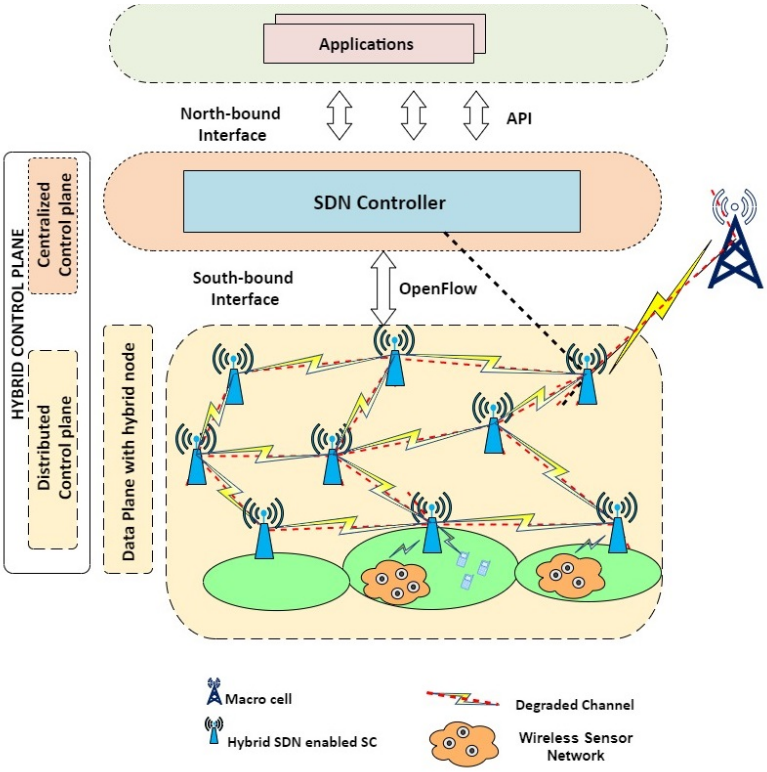

Figure 1. The proposed hybrid control plane architecture for DenseNet.

3.2. Hybrid Node Architecture

The high-level architecture illustrated in Figure 2 aims to quickly accommodate the changes in the control plane by combining both centralized and distributed control logic in the same node, thus creating a hybrid node as a data plane device. The hybrid node consists of the data plane forwarding pipe and the control logic to decide on the operation of the forwarding pipe of the network node. In what follows, we describe the architecture of the main components embedded in the network node.

3.2.1. Data Plane Forwarding Pipe

We adopted the Open Source Hybrid IP/SDN networking (OSHI) framework that was designed in refs. [19–21]. This framework allows nodes to concurrently run a distributed control plane and a centralized control plane. To attain this goal, the hybrid node embeds an SDN Capable Switch (SCS)—in our case, Open vSwitch, an IP-based forwarding engine (i.e., the one provided by the Linux kernel), and an IP routing daemon based on Quagga to calculate distributed routes (see Figure 2).

The SCS is connected to the physical network via the physical interfaces, while the IP forwarding engine is connected to the SCS via a set of internal virtual ports endowed in the SCS. We used Multiple Flow Tables (MFTs) [22] to separate: (i) flow rules that forward packets between SCS physical ports and internal virtual ports for having distributed operation and (ii) flow rules that are installed by the intervention of the SDN controller to handle centralized operation.

The main flow table embedded in the SCS allows for distinguishing between regular IP packets that are needed to be processed by the distributed control plane (i.e., Quagga routing daemon) and those needed to be processed by the SDN controller. VLAN (Virtual

LAN) IDs have been used to distinguish between packets that need to be processed by a distributed control plane and packets that have to be processed by SDN controller. In particular, the SDN controller is responsible for policy making, while packets come with VLAN IDs. On the other hand, packets without a VLAN ID are processed by IP routing daemon.

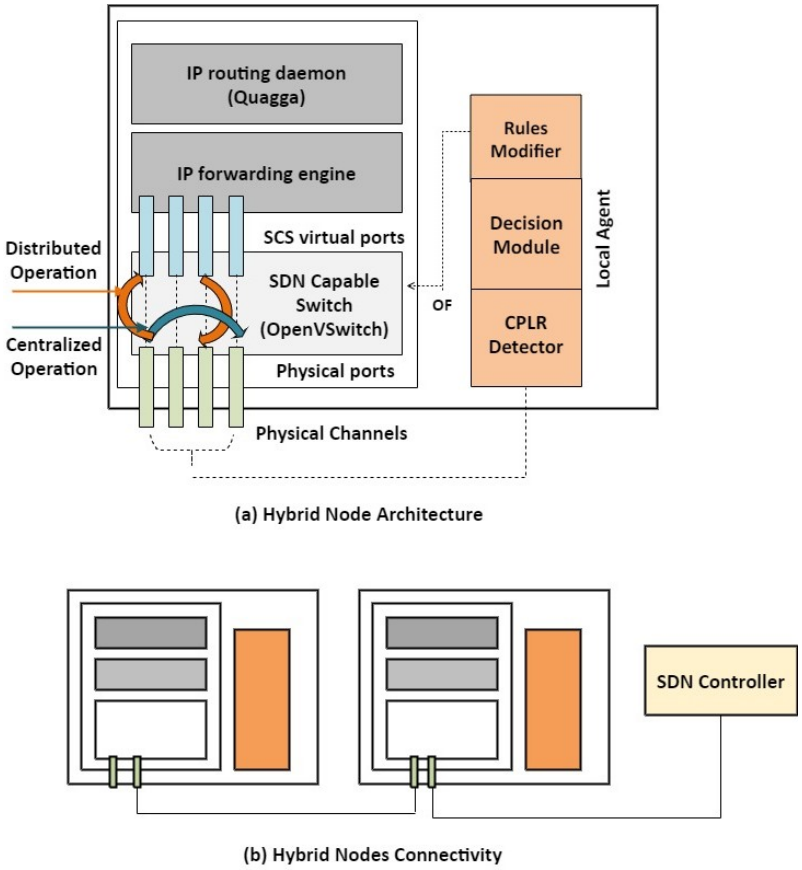

Figure 2. Hybrid node architecture with an integrated local agent.

Figure 3 describes the flow chart that shows the processing of packets in the SCS flow table depending on the operational mode (i.e., distributed or centralized). Packet processing starts at the main flow table, which is *Table-0*. In the case of centralized operation, the VLAN ID is not removed from the incoming packets, and the packets are directed to *Table-1*. If there is a matching rule for the packets, the corresponding action is taken against the packets. Otherwise, packets are forwarded to the SDN controller for policy making.

The policies that are set by the SDN controller are then installed into *Table-1* as flow entry rules. The subsequent packets are then forwarded according to the instructions set by the SDN controller. On the other hand, while distributed operation prevails in the network and is responsible for distributed policy making, the VLAN ID is removed from the packets, and the packets are directed to *Table-2*. The actions set that are integrated into *Table-2* are then executed to perform IP routing.

3.2.2. Local Agent

We included a local agent in the data plane node in order to enable centralized or distributed control depending on the reliability of the control communication channel between data plane devices and the centralized SDN controller. The local agent is composed of *(i) a CPLR detector (ii) a decision module, and (iii) a rules modifier*. The module *CPLR detector* is based on a monitoring framework that continuously infers the reliability of the control plane by periodically monitoring the status of the control communication channel. In this

paper, the metric is based on determining the packet loss ratio of the control communication channel.

The resulting metric is referred to as the Control Packet Loss Ratio (CPLR) (as well as the slope of the $CPLR$ vs. t curve), which determines the status of the control communication channel between the data plane node and the SDN controller. Depending on the reliability of the control communication channel, which is determined by the CPLR value of the links, as well as the CPLR trend, the decision module integrated into the data plane nodes (see Figure 2) decides activation of the distributed operation from the centralized operation and vice versa. In this sense, it embeds a control logic switching algorithm (see Section 3.3) that selects the mode of operation of network nodes by detecting trends that describe the quality of the control communication channels.

The prediction is based on the various measurements gathered from the centralized or the distributed control plane logic, which is active in a given data plane node. Under a high loss regime, the decision module triggers the action to perform network switching from centralized to distributed control plane operation. Then, the *rules modifier* of the local agent pushes some predefined rules to activate the distributed operation in the node that is decided by the decision module.

Though the network switching operation happens because of the failure of the centralized operation, the module *CPLR detector* of the local agent continues monitoring the control communication channel as well as the control plane. While the control plane as well as control communication channel performance becomes fair and the *CPLR detector* of the local agent characterizes the control channel as a reliable-enough medium by measuring the CPLR, the decision module again triggers the action to switch back network operation to a centralized mode.

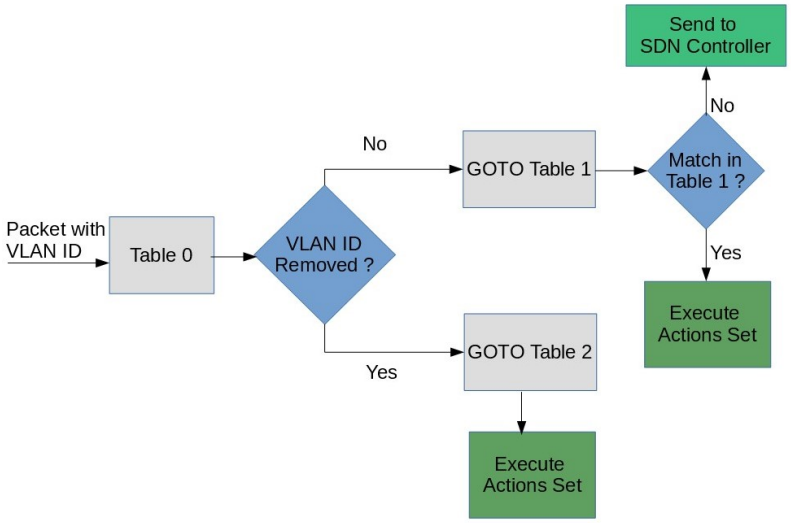

Figure 3. Packet processing in the SCS flow table.

3.2.3. Interaction between the Data Plane Node Forwarding Pipe, The Local Agent, and the Centralized SDN Controller

By default, packets are tagged with a VLAN ID to enable the use of a centralized control plane logic embedded in the SDN controller. In the following, we summarize the centralized operation.

1. When a packet with VLAN ID is received by a physical interface of the SCS, the SCS conducts a lookup in its flow tables to find a match for the current packet.
2. If a match is found in one of the flow tables, the packet is then forwarded according to the rule installed previously by the SDN controller into that flow table. Otherwise, the incoming packet is forwarded to the centralized SDN controller for the appropriate handling of the packet according to the policies defined. Section 3.2.4 recalls the regular operation of canonical SDN.

3. The centralized SDN controller installs the necessary OpenFlow rules to serve the current incoming packet. Thus, the forwarding data plane node simply follows the instructions set by the SDN controller to forward a packet.
4. The *CPLR detector* module of the local agent periodically monitors the control communication channel as well as the control plane by inspecting CPLR.

3.2.4. Canonical-SDN Operation

Let us briefly recall how Canonical-SDN operates in our setup and what kind of messages are exchanged between nodes and the SDN controller.

In the centralized SDN setup, when a new packet comes to the ingress port of a node (e.g., Open vSwitch), the node looks up in its flow tables for a flow entry match for the packet. If no match is found in the flow tables, the node sends the packet as a *PacketIn* message to the SDN controller using the *flow-miss* entry rule. Then, the SDN controller calculates the shortest path for the packet, and, using a *PacketOut* message, the SDN controller instructs the forwarding node to install flow rules into the flow table of the nodes. The forwarding nodes keep the rules for an infinite period unless the nodes get further instruction from the SDN controller to modify or delete the rules. With the subsequent flow of packets, the forwarding nodes simply forward the packet according to the rules installed previously with the intervention of the SDN controller.

In the case of a change of network topology, for instance, the link between two nodes goes down, the nodes that are affected instruct the SDN controller immediately about the impairment by sending *PortStatus* messages. In our case, when the SDN controller receives *PortStatus* messages, the SDN controller instructs, by means of *FlowMod* messages, the affected nodes as well as all the nodes in the affected path to delete all the previous rules from their flow tables except the *flow-miss* entry rule.

The *flow-miss* entry rule is not deleted because, using this rule, the forwarding node sends new packets to the SDN controller when a matching rule can not be found in the forwarding table of the node. When the forwarding nodes in the affected path receives instructions from the SDN controller to delete rules, the nodes perform deletion of rules according to the instructions of the SDN controller.

As the rules have been deleted from the flow table of the nodes, when a packet comes to the ingress port of the nodes, the nodes again send the packet as a *PacketIn* message to the SDN controller using the *flow-miss* entry rule. The SDN controller then sets a new policy for the incoming packets and instructs all the nodes in the new path to install new rules in their flow table. The nodes then forward subsequent packets according to the new rules.

All these control messages are fundamental for the operation of the network. Therefore, any loss affecting them may have noticeable consequences on network performance. The impacts of such losses as well as the operation of our proposed solution are evaluated in Section 4.

3.3. Control Logic Switching Algorithm

As a consequence of any impairment experienced by the control plane, due to either failure of the SDN controller or a degraded control channel, CPLR values will increase, and an anomaly will be inferred by the network node. Algorithm 1 describes the control logic switching algorithm. All state decisions have been explicitly reflected for the sake of clarity. This algorithm is periodically run in the local agent of each node. Each of these periods is referred to in Algorithm 1 with subindex $k \in \mathbb{N}$. The local agent will call this algorithm by providing the *current_state*, the measured CPLR (calculated by the *CPLR detector* module), and the slope calculated in the previous period (S_{k-1}) and the current one ((S_k)) and will obtain, as output, the operational mode to be configured, which will only be applied by the local agent if different from *current_state*.

There are various design criteria behind this algorithm. Since our goal is to stay as much as possible in centralized mode and only use the distributed mode as abackup

operational mode, the algorithm is more conservative (i.e., it attemtps to make sure) when switching from centralized to distributed than vice versa (lines #3–15). The rationale behind this is that the application of network-wide management policies is easier when operating in centralized mode due to easier programmability through the SDN controller APIs and functionality.

Furthermore, switching to distributed operation implies certain control message exchange and processing (Section 3.3.1). On the other hand, the algorithm will decide to switch to centralized mode in a greedier manner (i.e., based on slight improvements in network conditions), for the reasons explained above (lines #16 et seq.), as soon as network conditions improve.

To control its operation, there are two main parameters in the algorithm. First, $CPLR_{max}$ is the CPLR under which TCP retransmissions are enough to guarantee (even if delayed) the interaction between the SDN controller and the node, though, as observed in Section 4, there may be an impact in network performance. Second, the conservative behavior is represented by parameter δ, the value of the slope (S_k) of the curve CPLR vs. t in the current period k that, if exceeded, implies a noticeable increasing trend of CPLR (line #6).

Therefore, it is based on an increasing trend and not based on instantaneous CPLR values that the algorithm decides to switch to distributed operation. That is, at least it takes having an increasing slope above δ during two evaluation periods to switch (lines #6–7). However, if the instantaneous value is such that CPLR is above $CPLR_{max}$, which means that network performance cannot be guaranteed (even if with high losses), the switching decision is taken without waiting for evaluating the slope in the following period (line #13).

Algorithm 1 Pseudo code of the control logic switching algorithm.

Input: *current_state*, $CPLR$, S_{k-1}, S_k
Output: *next_state* ▷ centralized or distributed

```
1:  procedure DECIDE_STATE
2:      if current_state=centralized then
3:          if (CPLR < CPLR_max) then
4:              if S_k ≤ δ then                          ▷ If low CPLR increasing trend
5:                  next_state=centralized
6:              else
7:                  if S_{k-1} > δ then
8:                      next_state=distributed
9:                  else
10:                     next_state=centralized
11:                 end if
12:             end if
13:         else                                         ▷ CPLR ≥ CPLR_max
14:             next_state=distributed
15:         end if
16:     else                                             ▷ current_state=distributed
17:         if S_k >= 0 then
18:             next_state = distributed
19:         else
20:             if CPLR < CPLR_max then
21:                 next_state=centralized
22:             else
23:                 next_state=distributed
24:             end if
25:         end if
26:     end if
27:     return next_state
28: end procedure
```

When in distributed operation (lines #16 et seq.), the aim is to switch back to centralized when an improvement in network conditions is detected. This is coded in the algorithm as measuring a non-positive slope in this period (no need to wait for two periods, as above) (line #19), and the CPLR value is below the one that guarantees that the network can operate in centralized mode ($CPLR_{max}$) (line #20). Under other conditions, the algorithm decides that the problem persists and decides to stay in distributed mode.

3.3.1. Network Operation Switching: Centralized to Distributed

In the case of high loss conditions, when the CPLR of the control communication channel is increasing and the decision module detects that the CPLR curve is sloping upwards or the CPLR value reaches $CPLR_{max}$, the switching algorithm integrated into the local agent of the nodes takes the network operation switching decision from centralized to distributed mode. In order to perform network operation switching from centralized to distributed, the following steps are followed by the local agent:

1. The *rules modifier* module deletes the old rules, except the *flow-miss* entry rule, from the SCS flow tables that were installed with the intervention of the SDN controller, as this rule is important while network operation is again restored to centralized mode (see Section 3.3.2).
2. The *rules modifier* module installs new rules in the SCS flow table, which include an OpenFlow *POP_VLAN* action, aiming to remove the VLAN tag of the incoming packets to the ingress port of a node and to forward the incoming packets to the IP forwarding engine.
3. Incoming packets arriving at the SCS ingress port are then forwarded to the IP forwarding engine via internal virtual port, in order to process the packets through the IP routing daemon. In the IP routing daemon, a distributed routing protocol (in our case, Open Shortest Path First (OSPF)) makes the policies to route packets to their intended destination and send the packets to the SCS egress port via internal virtual port (see Section 3.2.1).
4. On the other hand, the *rules modifier* module of the local agent also updates the SCS flow table by adding another rule that pushes the VLAN tag again to the packets using an OpenFlow *PUSH_VLAN* action before the packets leave the SCS egress port and sends the packets toward the following hop toward the destination host. All these modifications (e.g., deletion and installation) of rules in SCS flow table are performed without the intervention of the SDN controller.

Whenever the network operation in the node has been switched to distributed mode, the node act as a legacy device where a distributed routing protocol (e.g., OSPF) ensures the policy making and routing of incoming packets. The SDN controller, in that case, does not have any influence over the nodes in the network because of control packet loss due to impairment in the control communication channel.

3.3.2. Network Operation Switching: Distributed to Centralized

During the distributed mode of operation, the module *CPLR detector* of the local agent continues measuring the CPLR (and its slope) of the control communication channel, and when the CPLR goes below $CPLR_{max}$ (and there is a non-increasing CPLR trend), the switching algorithm takes the network operation switching decision from distributed to centralized again. The local agent then restores centralized operation back by following the steps as follows:

1. The *rules modifier* module deletes the rules from the SCS flow table that includes *POP_VLAN* and *PUSH_VLAN* OpenFlow actions that have been installed by the local agent during network operation switching from centralized to distributed. When the control plane performance becomes fair again, the SDN controller starts receiving LLDP (Link Layer Discovery Protocol) messages and, by receiving LLDP messages, the SDN controller discovers the network topology again.

2. As when switching from centralized to distributed, all the flow rules installed by the SDN controller have been deleted, except the *flow-miss* entry rule, when the SCS ingress port receives a packet, it sends the packet to the SDN controller for policy making using the *flow-miss* entry rule, as there is no matching rule remaining in the SCS flow table. In this way, network operation switches back again to centralized mode and the SDN controller takes control of the network.

4. DenseNet-Hybrid SDN: Performance Evaluation

We consider a SC backhaul scenario (Figure 1) where the transport node associated with each SC is a hybrid node that connects heterogeneous networks (e.g., sensor networks, wi-fi networks, and mobile ad-hoc networks) to the aggregation point (e.g., the macro-cell base station) of a mobile network. We emulate such grid mesh networks of different topologies using Mininet [23] version 2.2.1. We also consider the in-band deployment of the SDN scenario where the SDN controller is connected via a gateway node. Ryu [24] was used as SDN controller to set the forwarding rules in network nodes.

We use TCLink [25] to emulate high capacity mm-Wave links (60GHz 802.11ad WiGig links) as wireless transport links and to set their link rate up-to 1 Gbps. The traffic flows from each SC are sent toward the aggregation point (e.g., macro base station), which in a realistic scenario may be co-located with a macro base station cell site and SCs deployed in lampposts, for instance. Multiple TCP flows were generated from different SCs by using iPerf [26], which, on average, resulted in a traffic rate of 50 Mbps each.

Table 1 explains the parameters used in our evaluations. Extensive evaluation campaigns under multiple network conditions and sizes allowed tuning the algorithm parameters ($CPLR_{max}$ and δ) to the values reflected in the table. Furthermore, 10 s. was found to present an appropriate trade-off between the processing overhead (much higher if evaluation periods were much shorter) yet being able to detect steady trends based on which the algorithm takes decisions.

Finally, the tested network sizes, CPLR values, and number of impaired links were selected to reflect networks with various complexities and a variety of operational conditions ranging from perfect to really bad, so that the algorithm could show its operation under stress and have its operational limits assessed. All tests were repeated 10 times, and their min, 25-percentile, median, average, 75-percentile, and max values are represented in the boxplots.

Table 1. The experimental parameters and their corresponding values.

Parameter	Value	Explanation
$CPLR_{max}(\%)$	15	CPLR value over which the network is not operational in centralized mode.
δ	0.5	Slope that if exceeded for two consecutive periods implies switching to distributed mode.
Period	10	Period (in s.) with which CPLR is evaluated by local agent of each node.
Network size	9, 16	Number of nodes in the grid.
CPLR (%)	5, 10, 15, 20, 25	Average CPLR values generated to test network performance.
Links impairments	0, 1, 2, 3, 4, 5	Number of link impairments (switching links up/down) randomly distributed through the network (yet guaranteeing having a connected graph).

4.1. Canonical-SDN Operation: Unreliable Condition

First, we conducted an experiment to observe the throughput and latency performance of canonical-SDN under unreliable conditions. To observe the impact of the unreliable control plane over data plane throughput and latency in a controlled way, we generated losses of control messages that were exchanged between the forwarding nodes and the

SDN controller. We used netem [27] in combination with TCLink to add impairments to all of the communication channels (including the channel that connects the SDN controller to the gateway node) in our emulated scenarios.

We also broke data plane links (e.g., the link between two nodes) randomly by maintaining a sequence of link down/up at arbitrary periods of our emulation time. We did so to emulate the extreme instability that could appear in a wireless channel, and, in this sense, it represents a quite complex situation to be handled by the control plane. As with data plane link failure, control messages (*PortStatus*, *FlowMod*, *PacketIn*, and *PacketOut* messages) [5] continue being exchanged between the SDN controller and data plane nodes in order to redirect these control messages to the SDN controller over the degraded channels. Moreover, in our evaluated scenarios, every 5 s, LLDP messages are exchanged between the SDN controller and the forwarding nodes; the SDN controller knows about the topology by receiving LLDP packets from the data plane nodes.

We performed evaluations with different network sizes under unreliable conditions, by injecting various flows in order to understand what were the CPLRs that allowed the control communication channel to operate (even if with difficulties) and those for which it was impossible in most repetitions. Figures 4–7 depict the data plane performance metrics (aggregated throughput and average latency) during unreliable control plane conditions of canonical-SDN for various network sizes (blue curve for three flows and red curve for five flows in the centralized case).

We injected three and five TCP flows in each topology and varied the CPLR of each link. We also broke four data plane links randomly during our emulation time. For both network sizes, the results reveal that the increase of CPLR caused degradation of the data plane throughput and substantial growth of the latency in the canonical-SDN scenario, which is enormous for a CPLR of 25% . In the evaluated scenario, during the presence of link impairments in the data plane, the affected nodes sent *PortStatus* messages to the SDN controller to inform the SDN controller about data plane link failure.

In this case, due to the presence of faulty control communication links between the SDN controller and the forwarding nodes, the control messages (i.e., *PortStatus*, *PacketIn*, *PacketOut*, and *FlowMod*) were dropped and could not be exchanged on time. Moreover, LLDP messages that were used by the SDN controller for discovering network topology, were also dropped. As the SDN controller and forwarding nodes used a TCP connection to maintain the communication between them, dropped packets were retransmitted again.

Therefore, for low CPLR values (e.g., 10%) throughput was maintained to an acceptable level through TCP retransmission of control messages. However, in the case of high values of CPLR (e.g., 25%), although dropped control packets were retransmitted again, the high loss of the control packets affected the restoration of a new path at the data plane, which rendered the network unusable, as the impact was the same as if the SDN controller had failed. The impact on the aggregated throughput and latency were also highly noticeable.

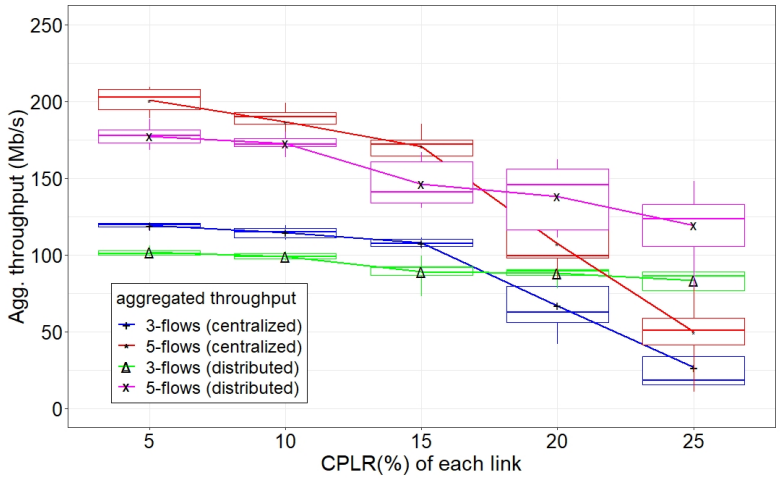

Figure 4. Data plane performance metric (aggregated throughput) comparison between centralized and distributed operations, during unreliable control plane conditions in a 9-node grid network.

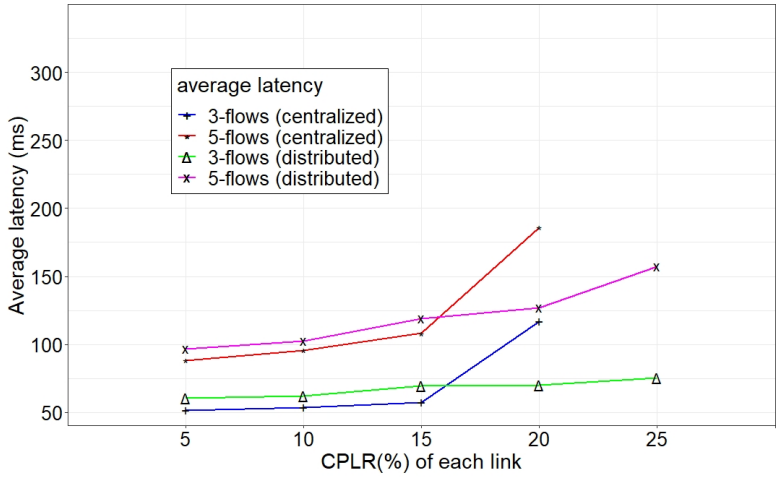

Figure 5. Data plane performance metric (average latency) comparison between centralized and distributed operations, during unreliable control plane conditions in a 9-node grid network.

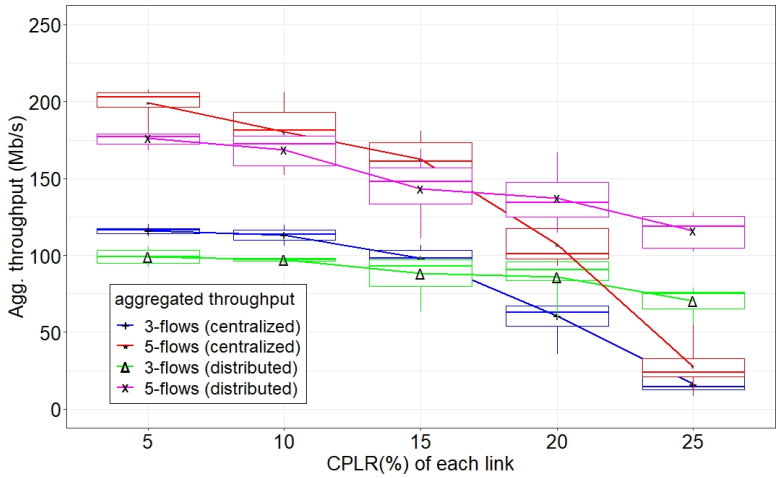

Figure 6. Data plane performance metric (aggregated throughput) comparison between centralized and distributed operations, during unreliable control plane conditions in a 16-node grid network.

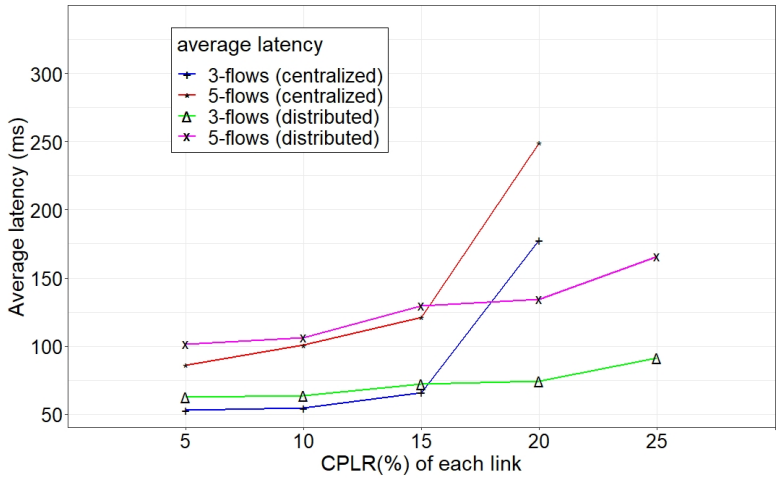

Figure 7. Data plane performance metric (average latency) comparison between centralized and distributed operations, during unreliable control plane conditions in a 16-node grid network.

To evaluate the impact of control message loss over data plane performance, we generated different numbers of link failures at the data plane with the intention of redirecting control messages to the SDN controller over lossy control communication channels. Figures 8–11 report the data plane performance of canonical-SDN while the network was experiencing an increasing number of data plane link impairments under a certain CPLR value. We repeated the experiments for different network topologies by injecting seven TCP flows in each topology.

We observe that, for both topologies, the aggregated throughput at the data plane substantially degraded, and the latency increased with the increasing number of broken links at the data plane. This was evaluated for various values of CPLR to assess its dependency on varying degrees of data plane link impairments. As CPLR increased, so did the restoration time due to control packet loss and the consequent retransmission of control packets between the network nodes and the centralized SDN controller. For both network topologies, for CPLR values of 25%, the degradation of the aggregated throughput at the data plane was remarkable, and the latency was substantially higher due to the high loss of control packets, which caused several retransmissions that affected the proper communication between the SDN controller and the forwarding nodes.

As latency is very high during a CPLR of 25%, for both cases, this is not represented in the figures. The network topology is a grid, and the connectedness of the graph is maintained, which may not be the case in a real deployment with lower nodal degrees of the graph. In this sense, the network performance presented in the figures may be considered the best possible case in terms of the path diversity.

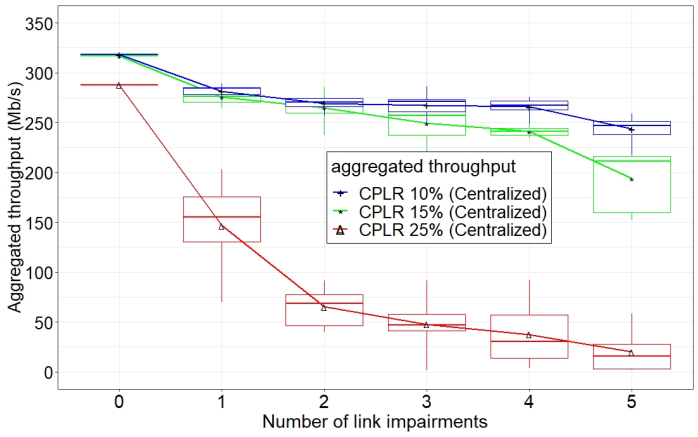

Figure 8. Aggregated throughput behavior of centralized network operation during data plane link failures for various times under unreliable control plane conditions in a 9-node grid network.

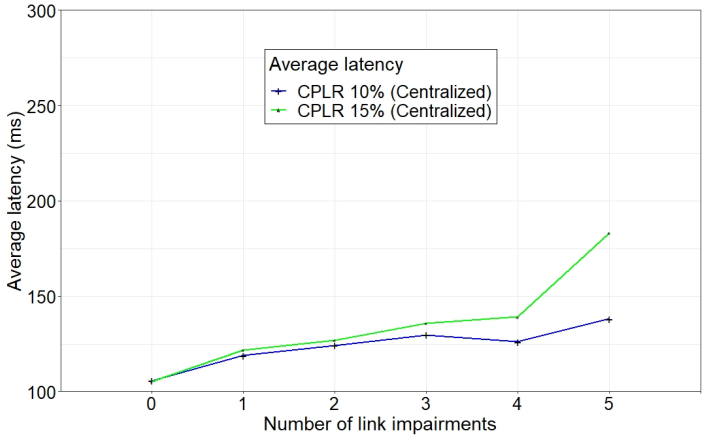

Figure 9. Average latency behavior of centralized network operation during data plane link failures for various times under unreliable control plane conditions in a 9-node grid network.

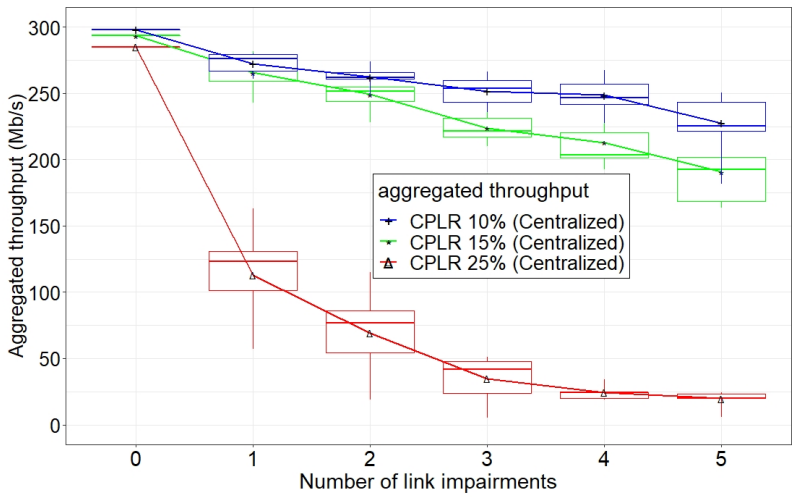

Figure 10. Aggregated throughput behavior of centralized network operation during data plane link failures for various times under unreliable control plane conditions in a 16-node grid network.

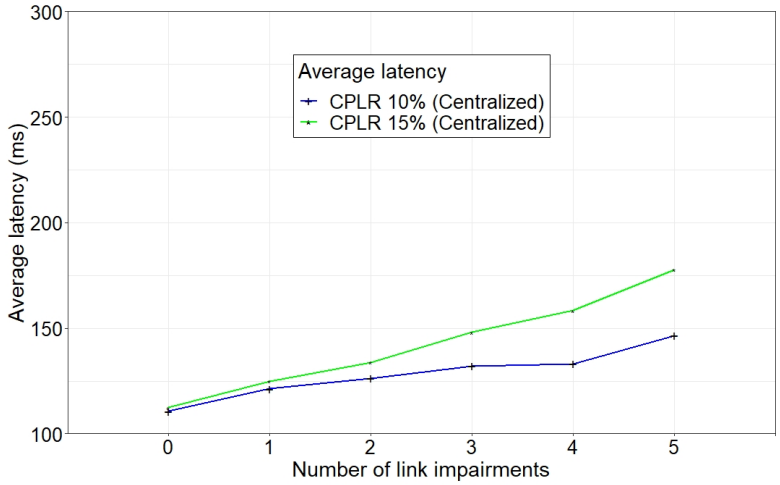

Figure 11. Average latency behavior of centralized network operation during data plane link failures for various times under unreliable control plane conditions in a 16-node grid network.

4.2. Distributed Operation: Unreliable Condition

We evaluated the same network topologies with the same number of TCP connections in the distributed network. We conducted our experiments in the distributed case where all the links were degraded communication links, and, in a controlled way, we increased the loss of control messages exchanged between two neighbor nodes. In the case of our evaluated distributed scenario, Hello and LSA (Link State Advertisement) messages are periodically exchanged among the neighbor nodes.

Every 2 s, Hello messages are exchanged between two neighbor nodes that confirm the availability of the neighboring nodes. On the other hand, every 5 s, LSAs are exchanged by the nodes in order to learn about the topology of the network. During any change in the network topology that may be caused by link failure or the unavailability of nodes, updated LSAs are exchanged by the nodes to learn about the changes in the topology.

In our evaluated scenario, we generated losses of these control messages in order to investigate the throughput and latency performance of distributed operation during unreliable conditions. In a way, we degraded the control communication channel by dropping control messages, and we also changed the network topology by breaking links in the network. We did so in order to exchange control messages during unreliable conditions.

When there is a change of topology (e.g., link failure happens), the nodes exchange the updated LSAs to learn about topology changes. However, due to the degraded control communication channels, some control messages are dropped, which, in turn, affects the network convergence time. The dropped control messages are retransmitted again; however, such losses affect the throughput and latency performance, as there is some delay in the reestablishment of routes caused by control packet loss.

From Figures 4–7, it can be observed that, in distributed network operation (green curve for three flows and purple for five flows in distributed case) and for low CPLR values (e.g., 5%), the aggregated throughput was higher, and the latency was lower because of lower control packet loss and also due to the lower retransmission rate of lost control messages. As CPLR increased, there was a degradation of the aggregated throughput while the latency went upward. With the increase of CPLR values, the loss of control messages increased, which also increased the rate of retransmissions. For this reason, and for higher values of CPLR (e.g., 25%), the aggregated throughput declined while the latency increased.

In another experiment, we investigated the performance of distributed operation during topology changes under unreliable conditions. In this way, for a certain value of CPLR, we changed the network topology several times by varying the number of broken links. As Figures 12–15 depict, a high number of link failures under the high CPLR regime (e.g., 25%) caused a decline in the aggregated throughput and an increase in the latency.

Due to a high loss of control messages, during topology changes for several times, the exchange of the updated LSAs between nodes suffered from losses, which caused more retransmissions and the consequent delay of control messages that affected the throughput and latency of the network.

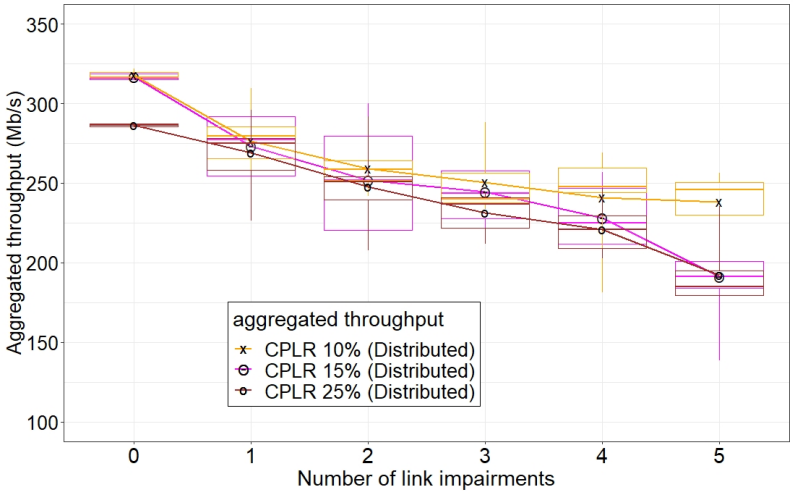

Figure 12. Aggregated throughput behavior of distributed network operation, during data plane link failures for various times under unreliable channel conditions in a 9-node grid network.

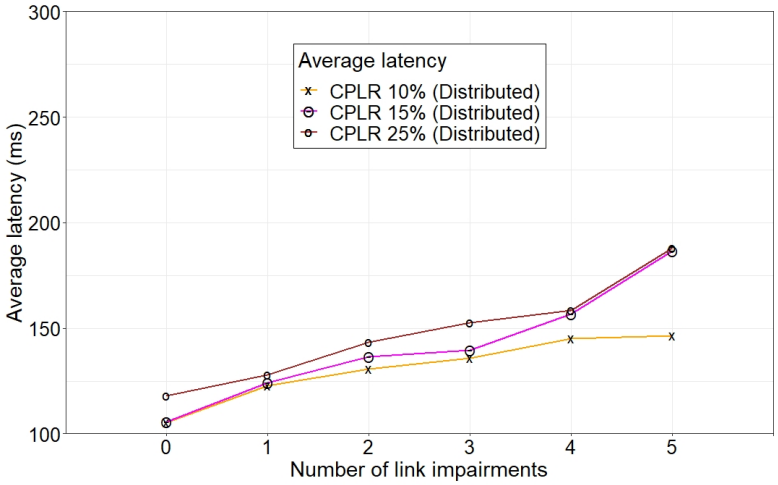

Figure 13. Average latency behavior of distributed network operation, during data plane link failures for various times under unreliable channel conditions in a 9-node grid network.

4.3. Performance Comparison: Canonical-SDN and Distributed Network

To observe the performance of centralized canonical-SDN and distributed operation under unreliable conditions, we generated controlled losses of: (1) control messages exchanged between the forwarding nodes and the SDN controller in the centralized mode and (2) messages exchanged between nodes that keep track of neighbor availability in the distributed mode of operation. From Figure 4–7, for both network sizes and for low CPLR values, centralized operation performed better (in terms of the throughput and latency) than distributed operation due to lower retransmissions, and the delay of control packets to reach the SDN controller is lower.

When the CPLR increased above 15%, distributed operation outperformed centralized operation by maintaining higher throughput and lower latency as higher retransmission of control packets was required for centralized operation, and also delay was incurred by the control packets. Moreover, Figures 8–11 illustrate that, for both network sizes and for high

CPLR values (e.g., 25%), while the topology of the network changed very often, centralized operation failed to operate correctly. On the other hand, distributed operation kept the network operational (see Figures 12–15).

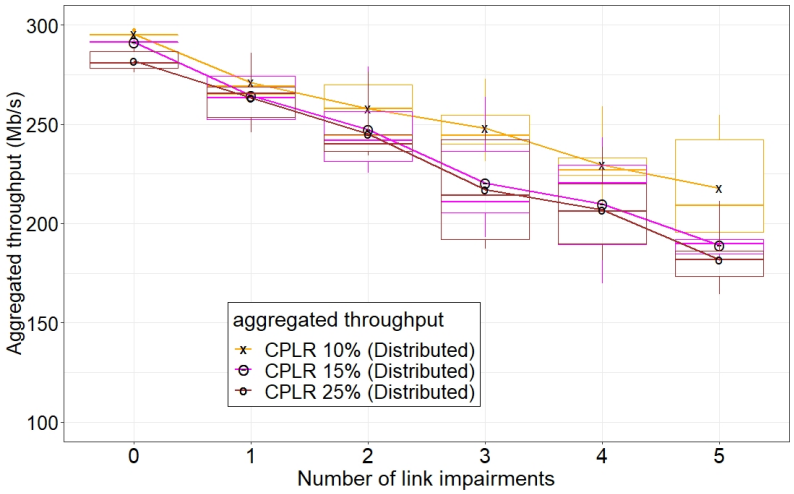

Figure 14. Aggregated throughput behavior of distributed network operation, during data plane link failures for various times under unreliable channel conditions in a 16-node grid network.

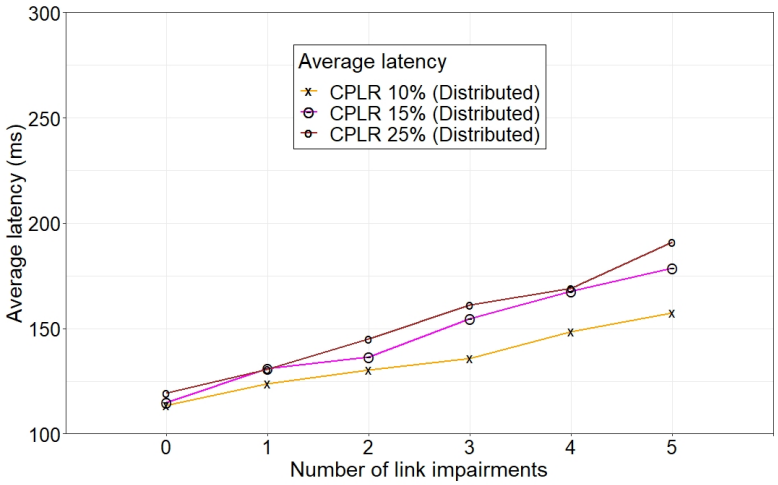

Figure 15. Average latency behavior of distributed network operation, during data plane link failures for various times under unreliable channel conditions in a 16-node grid network.

For low values of CPLR, the centralized operation performed well, as lower retransmissions and small delays were experienced by the control packets, and there was a lower convergence time as the network graph was already known by the SDN controller. However, for high CPLR values, due to the high loss of control packets, the exchange of control messages substantially affected (due to higher retransmissions and consequent delay) the TCP-based control communication between the SDN controller and forwarding nodes. For this reason, if the CPLR of the links increased, the performance of the centralized operation substantially degraded, as it depends on a centralized SDN controller, which becomes a single point of failure in high-loss conditions.

On the other hand, as LSAs are flooded, if a node fails to receive a copy of the LSA from a node due to the degraded channel, it might still receive a copy of that LSA from another node. In this way, the distributed link-state routing can converge during topology changes and shows better performance than the centralized operation under high control packet loss conditions.

4.4. Hybrid-SDN Operation

In the previous section, the performance evaluation of the canonical-SDN and distributed network during unreliable conditions has been analyzed for different network sizes and several injected flows. From our evaluated scenarios, by examining the performance of the centralized and distributed modes during unreliable control communication channel conditions, we characterized the switching point between centralized and distributed operation to the CPLR value of 15% and set it as $CPLR_{max}$.

From the previous section that below this point, centralized operation performed better, and, above it, distributed operation outperformed the centralized operation (see Figures 4–7). In this section, we quantify the gains that hSDN operation offers. In this direction, we emulate the same network topologies by injecting the same number of flows; however, the difference is that the forwarding nodes are hybrid nodes, where centralized and distributed operations coexist and the nodes can take autonomous switching decision between both modes.

4.4.1. Centralized to Distributed: Recovery Operation

Initially, the network was operated in centralized mode. The local agent that was integrated into nodes (see Figure 2) periodically monitored the CPLR of the links every 10 s. Then, we began increasing the CPLR, and when the CPLR of the links increased, the decision module integrated with the algorithm made a network operation switching decision, and the local agent performed switching from centralized to distributed operation, as described in Section 3.3.1.

From Figures 4–7, during high CPLR (e.g., 20%), the canonical-SDN showed lower throughput and higher latency due to high control packet losses, whereas Figures 16–19 report that hSDN—by switching the network logic to distributed—was able to obtain a throughput improvement of up to 28%, and the improvement was much higher when CPLR increased even more.

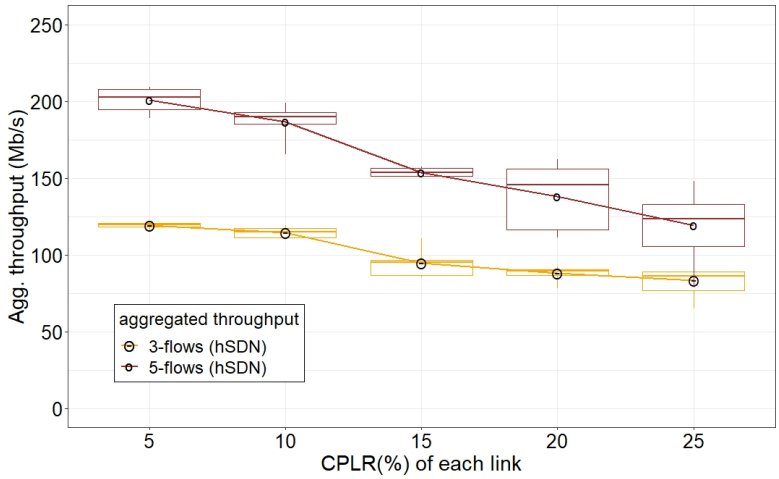

Figure 16. Data plane performance (aggregated throughput) of hSDN during data plane link failures for various times under unreliable control plane conditions in a 9-node grid network.

4.4.2. Distributed to Centralized: Good Control Plane Condition

During unreliable control plane conditions, the local agent integrated into the nodes performed network operation switching and also monitored the control communication channel periodically. As soon as the channel condition became good, which was determined by measuring the CPLR, the local agent again performed network operation switching from the distributed to centralized mode by following the procedure illustrated in Section 3.3.2. To illustrate the operation of the control logic switching algorithm, Figure 20 depicts the data plane performance duringthe network mode of operation switching (i.e., centralized-

distributed switching back and forth) in our hSDN approach by measuring the CPLR value of the control communication channel.

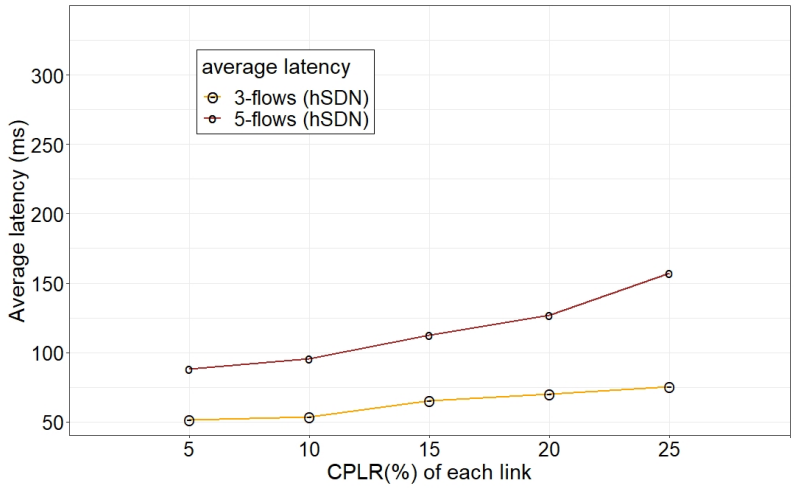

Figure 17. Data plane performance (average latency) of hSDN during data plane link failures for various times under unreliable control plane conditions in a 9-node grid network.

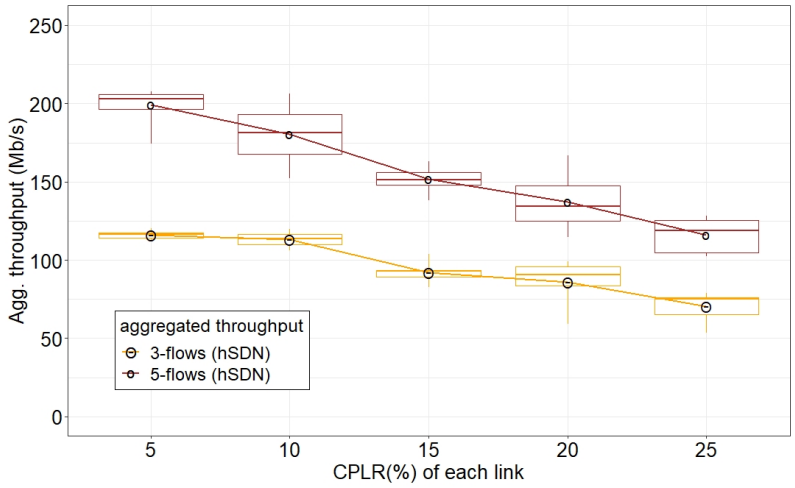

Figure 18. Data plane performance (aggregated throughput) of hSDN during data plane link failures for various times under unreliable control plane conditions in a 16-node grid network.

Initially, all the nodes in the 16-node grid network were in centralized mode. We began increasing impairments to the links and, at around 60 s, a high CPLR was measured, and the decision module of the local agent made a decision based on the integrated algorithm to perform network mode switching and performed switching to the distributed operation. At around 120 s, the performance of the control communication channel became fair (i.e., low loss), and, as a consequence, the CPLR became low, and the local agent restored the centralized operation based on the decision taken by the decision module. By doing so, the SDN controller gained control of the network, and the network was managed in a centralized fashion.

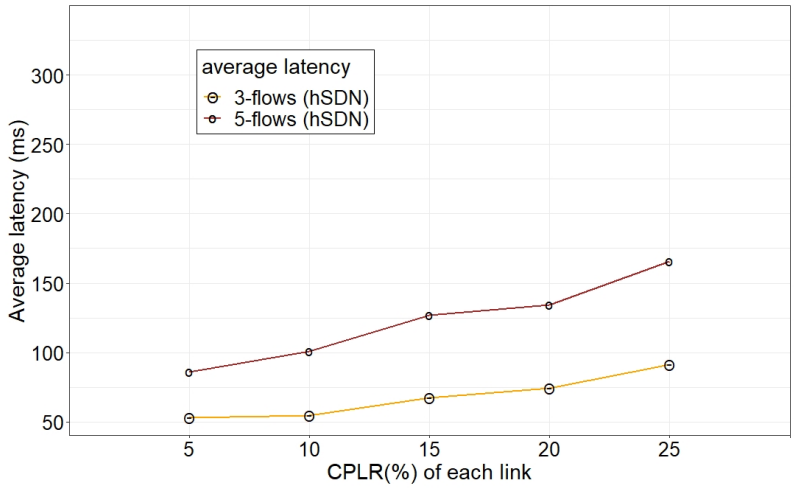

Figure 19. Data plane performance (average latency) of hSDN during data plane link failures for various times under unreliable control plane conditions in a 16-node grid network.

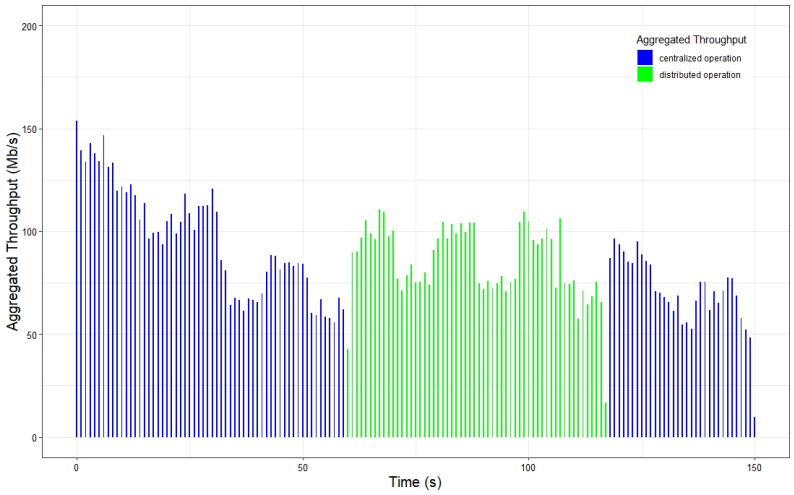

Figure 20. Aggregated throughput behavior over time during network operation switching in hSDN.

5. Comparison of hSDN Schemes

Our proposed hSDN approach brings novelty in the way the control plane failure is handled. We considered the metric Control Plane Loss Ratio (CPLR) to investigate reliability of control communication channel and also argue that the investigation should be done from node points of view. By doing so, failure of the controller, control communication channel, and even degraded control communication channel can be detected, which brings huge advantages in network management. Moreover, our approach introduces a network switching algorithm that predicts the quality of the control communication channel by detecting trends of the control packet loss curve (i.e., slope of the CPLR vs. t curve) and makes an autonomous decision to switch the network operation.

A comparative analysis between the proposed approach and the existing approaches is presented in Table 2. The comparison was made based on control plane reliability. We considered the following fault detection criteria: A-SDN controller failure, B-control communication channel failure, and a C-degraded control communication channel. The "+" sign indicates the presence of the fault detection criteria and scores 1 point. The "−" sign defines the absence of the criteria and no score (i.e., 0 (zero) point).

Table 2. Comparative analysis of hSDN schemes.

Schemes	A	B	C	Score
HyperFlow [15]	+	-	-	1
ResilientFlow [16]	-	+	-	1
Ravana [17]	+	-	-	1
wmSDN [18]	+	-	-	1
Proposed	+	+	+	3

6. Discussion

Generally, the centralized SDN is assumed to have an uninterruptible communication between forwarding devices and the SDN controller. However, in a multi-hop wireless network scenario, the performance of centralized SDN may degrade as the in-band channels are responsible for carrying control packets from forwarding devices to the SDN controller and the wireless channels are more likely to be experiencing impairments due to environmental conditions. In such a scenario, the performance (in terms of data plane throughput and latency) of SDN may degrade or even become nonoperational during critical conditions (i.e., high control packet loss), which was analyzed in our work.

Apart from the throughput and latency metrics, the Available Bandwidth (ABW) [28] measurement is one of the significant metrics in SDN to obtain information about the current load on the links as well as on the network. In the SDN scenario, to measure ABW, the controller periodically performs polling of the measurement report from the forwarding devices using *PortStatusReq* messages. As presented in [28], while delay increases in communication between the controller and the forwarding devices, the ABW estimation error also increases, which has an adverse impact on the path selection for any services.

For in-band SDN deployment in a wireless multi-hop scenario where control packets are sent over the wireless links, due to environmental conditions, the communication between the controller and the forwarding devices may be hampered. The degraded control communication link may incur delay on control packets or even the loss of control packets, which was investigated in our work. Therefore, the loss of control packets (e.g., *PortStatusReq*) may have adverse impacts on ABW measurements.

7. Conclusions and Future Work

Since the SDN paradigm generally assumes full reliability of the control plane, this paper set its goal to evaluate what happens when this is not the case, and we also evaluated a possible solution. This scenario is becoming more likely in 5G when deployed at higher frequencies through massive small cell deployments served through wireless transport links. Given the criticality of control plane message losses or SDN controller failure due to network performance, we quantitatively evaluated the impacts on overall network performance as well as the improvements achieved when a hSDN approach is deployed.

By detecting control plane unreliability conditions, this scheme decided when to switch from centralized SDN operation to a distributed one. This way of working resulted in a substantial aggregated throughput improvement of the flows traversing the network (e.g., around 28% for CPLR = 20% and much more for higher CPLR) at the cost of a slight increase in the average latency in the evaluated scenarios. However, the latency gap between the optimal operation (centralized in good network conditions and distributed in bad network conditions) decreased as the network conditions worsened (i.e., a higher CPLR and more links experiencing problems).

Based on these results, we conclude that such hSDN approaches are promising in SDN-based deployments over unreliable transmission media. Our future work plans to explore to what extent and under what conditions Artificial Intelligence (AI) and Machine Learning (ML)-based techniques can improve hSDN network operation without adding much complexity to the nodes.

Author Contributions: M.O. was responsible for conceptualization, methodology, formal analysis, investigation, designing experiment and writing original draft. J.M.-B. was responsible for supervision, review and editing the article. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by Spanish MINECO grant TEC2017-88373-R (5G-REFINE) and also the research leading to these results was supported by Generalitat de Catalunya grant FI-DGR 2015 and 2017 SGR 1195.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data underlying this article will be shared on reasonable request from the corresponding author.

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.

References

1. Robson, J. Small cell backhaul requirements. In *NGMN White Paper*; NGMN Alliance: Frankfurt, Germany 2012; pp. 1–40.
2. Small Cell Millimeter Wave Mesh Backhaul. In *InterDigital White Paper*; InterDigital, Inc.: Wilmington, DE, USA, 2013.
3. Small Cells Drive Microwave Backhaul Boom. Small Cells 2019-2026. In *RAN Research Market Report*; Rethink Technology Research: Bristol, UK, 2020.
4. ETSI NFV. Report on SDN Usage in NFV Architectural Framework. In *ETSI GS NFV-EVE 005 v1.1.1*; Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG): Sophia Antipolis, France, 2015.
5. Open Networking Foundation. OpenFlow v1.3. Available online: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf (accessed on 19 August 2021)
6. Núñez-Martínez, J.; Baranda, J.; Pascual, I.; Mangues-Bafalluy, J. WiseHAUL: An SDN-empowered Wireless Small Cell Backhaul testbed. In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Coimbra, Portugal, 21–24 June 2016. [CrossRef]
7. Mangues-Bafalluy, J.; Baranda, J.; Landi, G.; Núñez-Martínez, J.; Casellas, R.; Chundrigar, S.B.; de la Oliva, A.; Mourad, A.; Talat, S.T.; Chiasserini, C.F.; et al. Experimental framework and evaluation of the 5G-Crosshaul Control Infrastructure. *Comput. Stand. Interfaces* **2019**, *64*, 96–105. [CrossRef]
8. 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]
9. Kim, H.; Schlansker, M.; Santos, J.R.; Tourrilhes, J.; Turner, Y.; Feamster, N. Coronet: Fault tolerance for software defined networks. In Proceedings of the 2012 20th IEEE international conference on network protocols (ICNP), Austin, TX, USA, 30 October–2 November 2012; pp. 1–2. [CrossRef]
10. Kuźniar, M.; Perešíni, P.; Vasić, N.; Canini, M.; Kostić, D. Automatic failure recovery for software-defined networks. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 16 August 2013; pp. 159–160.
11. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. Enabling fast failure recovery in OpenFlow networks. In Proceedings of the 2011 8th International Workshop on the Design of Reliable Communication Networks (DRCN), Krakow, Poland, 10–12 October 2011; pp. 164–171. [CrossRef]
12. ONOS Project. Distributed Mode. Available online: https://wiki.onosproject.org/display/ONOS/Distributed+Operation (accessed on 19 August 2021)
13. Vissicchio, S.; Vanbever, L.; Bonaventure, O. Opportunities and research challenges of hybrid software defined networks. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 70–75. [CrossRef]
14. Osman, M.; Núñez-Martínez, J.; Mangues-Bafalluy, J. Hybrid SDN: Evaluation of the impact of an unreliable control channel. In Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany, 6–8 November 2017; pp. 242–246. [CrossRef]
15. Tootoonchian, A.; Ganjali, Y. HyperFlow: A distributed control plane for OpenFlow. In Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, San Jose, CA, USA, 27 April 2010; p. 3.
16. Omizo, T.; Watanabe, T.; Akiyama, T.; Iida, K. ResilientFlow: Deployments of distributed control channel maintenance modules to recover SDN from unexpected failures. *IEICE Trans. Commun.* **2015**, *99*, 1041–1053. [CrossRef]
17. Katta, N.; Zhang, H.; Freedman, M.; Rexford, J. Ravana: Controller fault-tolerance in software-defined networking. In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, Santa Clara, CA, USA, 17–18 June 2015; p. 4.

18. Detti, A.; Pisa, C.; Salsano, S.; Blefari-Melazzi, N. Wireless Mesh Software Defined Networks (wmSDN). In Proceedings of the 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Lyon, France, 7–9 October 2013; pp. 89–95. [CrossRef]
19. Salsano, S.; Ventre, P.L.; Prete, L.; Siracusano, G.; Gerola, M.; Salvadori, E. OSHI-Open Source Hybrid IP/SDN networking (and its emulation on Mininet and on distributed SDN testbeds). In Proceedings of the 2014 Third European Workshop on Software Defined Networks, Budapest, Hungary, 1–3 September 2014; pp. 13–18. [CrossRef]
20. Salsano, S.; Ventre, P.L.; Lombardo, F.; Siracusano, G.; Gerola, M.; Salvadori, E.; Santuari, M.; Campanella, M.; Prete, L. Hybrid IP/SDN networking: Open implementation and experiment management tools. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 138–153. [CrossRef]
21. Open Source Hybrid IP/SDN Networking (OSHI) Home Page. Available online: http://www.netgroup.uniroma2.it/twiki/bin/view/Oshi (accessed on 19 August 2021)
22. *The Benefits of Multiple Flow Tables and Ttps*; Technical report, ONF Technical Report; Open Networking Foundation: Menlo Park, CA, USA 2015.
23. Mininet. An Instant Virtual Network on Your Laptop. Available online: http://www.mininet.org/ (accessed on 19 August 2021)
24. RYU—Component-Based Software-Defined Networking Framework. Available online: https://www.ryu-sdn.org/ (accessed on 19 August 2021)
25. TCLink in Mininet PYthon API Reference Manual. Available online: http://mininet.org/api/classmininet_1_1link_1_1TCLink.html (accessed on 19 August 2021)
26. iPerf—The Ultimate Speed Test Tool for TCP, UDP and SCTP. Available online: https://www.iperf.fr/ (accessed on 19 August 2021)
27. netem—Network Emulation. Available online: https://wiki.linuxfoundation.org/networking/netem (accessed on 19 August 2021)
28. Megyesi, P.; Botta, A.; Aceto, G.; Pescapé, A.; Molnár, S. Challenges and solution for measuring available bandwidth in software defined networks. *Comput. Commun.* **2017**, *99*, 48–61. [CrossRef]