Self-healing Topology Discovery Protocol for Software Defined Networks

Leonardo Ochoa-Aday, *Student Member, IEEE*, Cristina Cervelló-Pastor, *Member, IEEE* and Adriana Fernández-Fernández, *Student Member, IEEE*

***Abstract*—This letter presents the design of a self-healing protocol for automatic discovery and maintenance of the network topology in Software Defined Networks (SDN). The proposed protocol integrates two enhanced features (i.e. layer 2 topology discovery and autonomic fault recovery) in a unified mechanism. This novel approach is validated through simulation experiments using OMNET++. Obtained results show that our protocol discovers and recovers the control topology efficiently in terms of time and message load over a wide range of generated networks.**

***Index Terms*—topology discovery, protocols, software defined networks, fault tolerance, network management.**

I. Introduction

NETWORK topology discovery represents a description of the physical infrastructure in a network. This description contains information about the connectivity, latency and link capacity between the network devices at the low level (i.e. Data Link layer). In Software Defined Networks (SDN), the SDN controller (SDNC) collects the topology information from the data plane and maintains an abstract view of the entire network [1]. This topology discovery (TD) service is crucial for the proper functioning of topology-aware network applications (i.e. network configuration, traffic engineering, attack detection, load balancing, etc.) [2]. Although discovering the network topology is an essential service for SDN-managed networks, there is no official standard that defines a TD mechanism in SDN [1]. Therefore, most of current SDNCs (i.e. Beacon, NOX, Ryu, Floodlight, among others) [2] implement conventional TD mechanisms based on the southbound protocol OpenFlow [3].

Different mechanisms can be designed to implement the TD service in SDN. However, OpenFlow-based TD mechanisms need a preconfigured network identifier (i.e. IP address) on each forwarding device [3]. Otherwise, SDNCs will not be able to discover the OpenFlow switches in the network. Additionally, these mechanisms also require a previous knowledge of network devices' characteristics such as number of active ports and MAC addresses [4]. This information is requested by SDNCs after the establishment of an initial control connection with each device. Current SDNCs cannot implement any conventional TD mechanism without this requested information.

After discovering the network topology, the control plane needs a survivability strategy to guarantee its reliability at all times. In that direction, we propose a self-healing technique to boost the control plane resilience in SDN-managed environments without overburdening the SDNC performance.

The widespread adoption of SDN in heterogeneous and failure-prone deployments (i.e. data centers, clouds, etc.), has brought a general interest in providing SDN with the self-healing attribute. Although autonomic management is agreed as the next generation of management [5], we identify lack of proposals to integrate autonomic attributes (e.g. self-healing) into SDN. To the best of our knowledge, our protocol is the first one in proposing a unified mechanism for discovering the physical topology and providing native fault recovery in the control plane without intervention of the SDNC. Therefore, this letter proposes the design and simulation of a self-healing topology discovery (SHTD) protocol in order to improve the current TD service in SDN environments.

The main contributions and novelty of the proposed protocol are summarized as follows.

1) A prior topology knowledge or specific network configurations are not required to discover the network topology before the initial control connection.
2) Recovery of the network from node, link or SDNC failures autonomously and stably, by only taking actions at the forwarding level.
3) Integration of both features (i.e. topology discovery and fault recovery) as an enhanced mechanism.

The remainder of this letter is organized as follows: In Section II, we describe the proposed protocol in detail. Section III discusses the performance of the proposed protocol through simulations. Finally, we make some conclusions in Section IV.

II. Self-healing Topology Discovery Protocol

The SHTD protocol discovers and maintains an accurate network view integrating two modules. These modules are the following: the layer 2 topology discovery mechanism and the autonomic fault recovery mechanism. This protocol can be implemented in a domain with multiple SDNCs through a software agent running in each network device. The control plane can be interconnected with a plurality of network devices through different transmission media. Although in this work SDNCs use an in-band control scheme, this approach can also be implemented using out-of-band connections.

The global view of the network is provided by SDNCs, after running the SHTD protocol. Unlike conventional mechanisms [4], [3], the proposed protocol uses layer 2 messages to create a control tree rooted at SDNCs and it discovers the network topology before the establishment of the initial control

The authors are with the Department of Network Engineering, Universitat Politècnica de Catalunya, Esteve Terradas, 7, 08860, Castelldefels, Spain, e-mails: {leonardo.ochoa, cristina, adriana.fernandez}@entel.upc.edu.

This work was supported by the Ministerio de Economía y Competitividad of the Spanish Government under project TEC2016-76795-C6-1-R and AEI/FEDER, UE.

connection. Once the control tree is created, each network device can establish a secure control channel to the SDNCs. A similar approach has been considered in [6], [7]. However, these works do not provide any procedure or technique to recover the control tree after a network failure.

A. Layer 2 Topology Discovery Mechanism

The TD mechanism is initialized by each SDNC sending a topoRequest message. The propagation of this multicast message creates a control tree topology rooted at the SDNCs for collecting network state data. Moreover, this control tree distributes the management of the physical infrastructure among several SDNCs.

Besides the SDNC, nodes have one of the four roles: non-discovered, leaf, v-leaf or core. A node is non-discovered until it receives a topoRequest message from another node, leaf nodes are the external nodes in the network and a node is v-leaf when all of its downstream ports are connected to leaf or v-leaf nodes. The remaining nodes are denoted as core nodes.

Additionally, each active port has one of four states related to the control tree: standby, parent, child or pruned. Each port has a state machine that tracks and governs its current state during the control tree creation.

- A standby port is an active port in the node that is not used in the control tree.
- A parent port is an upstream port in the control tree, which has first received the topoRequest message. Each node cannot have more than one parent port.
- A child port is a downstream port which has received an echoReply message as part of the control tree.
- A pruned port is a child port which has received a topoReply message indicating that it is connected to a leaf or v-leaf node.

Initially, each node in the network is in the non-discovered state, waiting for a topoRequest message from a SDNC or another node. Algorithm 1 shows the mechanism, for a given node v, after receiving the topoRequest message. When node v receives the topoRequest from a node u, it sends a one-hop echoReply message to node u. This automatic reply enables the measurement of the Round Trip Time (RTT) in each network link. Moreover, it contains an association bit which is used by node v as a join message to announce to its neighbours whether it has joined one of them in the control tree. If node v receives the topoRequest by a standby port, it confirms the association in the echoReply, sets the parent state to the incoming port p and forwards the topoRequest to all the ports except the incoming port. By contrast, if the topoRequest arrives at a non-standby port, node v denies the association in the echoReply and discards the message. In this way, node v performs an implicit mechanism to detect and avoid loops.

As the tree is being created, each node periodically sends its neighbourhood data through the parent port using a topoReply message. This cyclic process is asynchronously started by the leaf nodes. Given their topological nature, if the parent ports of leaf and v-leaf nodes fail, they cannot provide an alternative control path to the SDNCs. Hence, a one-bit field is used in the topoReply message in order to prune their neighbouring port. In this way, unnecessary topoRequest messages are not forwarded to them in the control tree. This is critical to achieve minimal communication overhead in the proposed SHTD protocol. Meanwhile, core nodes aggregate the topology data from their child ports. Once they have received information through all their child ports, they send it towards the SDNCs. As a result, topoReply messages are gathered at the SDNCs, which receive at most an aggregated message from each of their active interfaces.

Algorithm 1 topoRequest message forwarding

```
1: Node v receives topoRequest from node u by port p
2: if p.state = Standby then
3:     Send echoReply to node u          ▷ association bit set
4:     StateMachine (p)
5:     Send topoRequest for all ports except p
6: else
7:     Send echoReply to node u          ▷ association bit clear
8:     Discard topoRequest
9: end if
```

B. Autonomic Fault Recovery Mechanism

By definition, autonomic networks comprise two major entities: the managed components and the autonomic manager [5]. In this context, both elements are identified as follows:

- *Managed Components*: Represented by the set of network devices which supports the SHTD protocol. Each managed component includes sensors to monitor the state of its neighbouring links.
- *Autonomic Manager*: Coupled within each SDNC, it is responsible for monitoring the managed components, analysing their collected data and optimise the network performance in order to better accomplish its high-level objectives (i.e. native control plane robustness).

The proposed autonomic mechanism is implemented by the network devices through topoUpdate messages in the data plane. When a network device detects a port failure (i.e. neighbour's connectivity breaks down), the managed component executes specific actions depending on the state of disrupted ports. Failures on standby, pruned or child ports are automatically informed to SDNCs with no changes in the upstream control tree. Please note that each child port is connected to a parent port of downstream nodes. On the other hand, if a parent port fails, the node have to autonomously recover its control connectivity by taking actions at the forwarding level only (i.e. with no SDNC interventions).

Fig. 1 shows an example of the SHTD protocol basic operation. For this sample topology, we consider two SDNCs connected to 8 switches through links that have the same unitary delay. This multi-SDNC platform can be deployed in the cloud using a secure and resilient architecture. After both SDNCs run the protocol, the control tree topology and its distribution of switches among SDNCs is depicted through colours and shapes in Fig. 1(a). For instance, if N2 fails as it is shown in Fig. 1(b), N6 and N7 have to send a topoUpdate message for all active ports, except the pruned ports, in order to find new control paths towards the SDNCs.

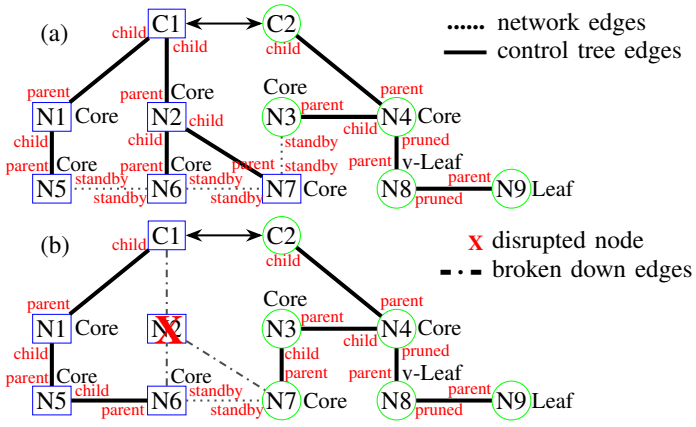

Fig. 1. Example of SHTD protocol operation.

To avoid propagation loops, each topoUpdate contains an ID field with the disrupted port ID, which can be generated from the MAC address of this port, as proposed in [3]. Thus, each port is uniquely identified in the network. When a topoUpdate is received, stored ID values are compared to the one received in the message. If a match exists, then the topoUpdate is discarded. Otherwise, this message is forwarded for all ports except incoming or pruned ports, in order to avoid forwarding unnecessary topoUpdate messages to nodes that cannot be used to recover the control tree topology.

To clearly illustrate what happens, Fig. 2 shows the message sequence for the proposed fault recovery mechanism when N6 detects its parent port down. As explained above, N6 sends one topoUpdate message to N5 and N7, simultaneously. This message is also forwarded from N7 to N3. Nodes that have its parent port active (i.e. N5 and N3), instead of forwarding the topoUpdate, send a replyUpdate message. This replyUpdate is sent one-hop back through the path followed by the received topoUpdate message, creating a new way for N6 to reach SDNCs. Upon receiving the first replyUpdate -sent by N5-, N6 sets parent state to the incoming port and it automatically sends a modified topoReply message to N5. Afterwards, following replyUpdate messages will be discarded (e.g. the one from N7). The modified topoReply message contains the association bit set, which is used by N6 to announce to N5 that they are already joined in the recovered control tree topology. Hence, N5 should update its port status as well as informs its SDNC (i.e. C1) with the updated topology data.

Once the connectivity is recovered in the forwarding plane, the recovered control topology can be optimized by the SDNCs. To achieve this, the autonomic managers, aware of the network knowledge contained in several control modules (such as network topology and statistics, routing policies, etc.), can better diagnose the problem [8]. As a result, each SDNC may change the overlay control topology in order to ensure the requirements of the supported network applications (e.g. load balancing, traffic engineering, monitoring, etc.).

C. Benefits of SHTD Protocol

The benefits of adopting the SHTD protocol are many-fold. First, the mechanism uses the least possible amount of

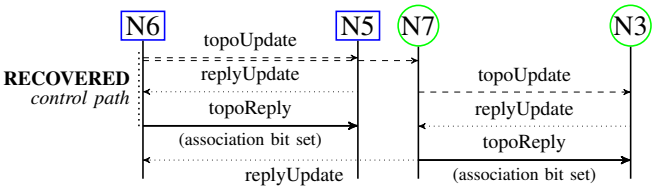

Fig. 2. Message flows for the autonomic fault recovery mechanism in N6.

messages (i.e. minimal communication overhead) and each message has a small size. Therefore, it is easy to implement and yet efficient. Second, echoReply messages enable accurate latency measurements within the network abstract view at SDNCs. Third, the proposed fault recovery scheme enables network devices to stably recover the network from "broken" states with very low recovery times, few packet loss and reduced memory requirements at network devices. Finally, the proposed SHTD protocol enables a survivability strategy to ensure the control plane reliability, as long as one SDNC is still reachable in the network.

III. Simulation and Results

We have implemented the SHTD protocol from scratch in OMNET++ network simulator, by its suitability to study realistic large-scale scenarios. To evaluate the performance of our solution across varying connectivity degree, we generated three sets of networks. Each network family was generated using an underlying topology from SNDlib [9]. Specifically, we select three network graphs representative of different scales, namely DfnBwin (10 nodes, 45 links, i.e., full mesh), NobelGermany (17 nodes, 26 links) and Zib (54 nodes, 80 links). Topologies that belong to NobelGermany and Zib based sets have been constructed as scale-free networks using a power-law node degree distribution with the same degree exponent as of the original network (NobelGermany: 3.06 and Zib: 3). This was a result of using the static Barabási-Albert model (i.e. keeping the original number of nodes and links). On the other hand, all networks based on DfnBwin have a full mesh topology. Different link latencies for each network family were randomly generated considering the mean and standard deviation values of the original topology used as master. For SDNC placements, we use the most central nodes in each topology based on the closeness centrality. All simulation results include their respective 95 % confidence intervals (CI) in the plots, based on Student-t distribution.

Fig. 3 shows the performance evaluation of the SHTD protocol for different key metrics. Each family size was determined after restricting the margin of error of the indicated average values to less than 3 % on each simulation instance. Specifically, DfnBwin, NobelGermany and Zib based sets are composed of 500 generated networks. For the computation of the presented time values, we consider the propagation latencies and the switch processing delays, as given in [10] for NetFPGA implementations. Furthermore, we use a Bidirectional Forwarding Detection (BFD) scheme to detect link losses instead of path failures, as proposed in [11].

In Fig. 3(a), we measure the average discovery time required by a number of SDNCs to discover the overall network

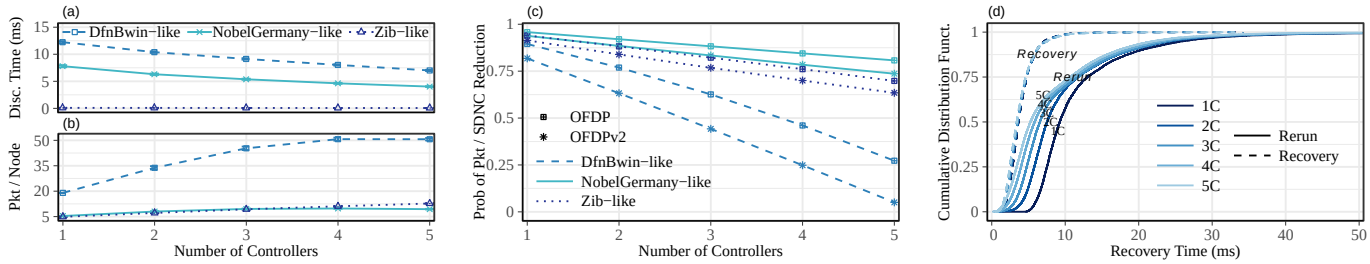

Fig. 3. Performance evaluation of SHTD protocol.

topology. The proposed discovery mechanism do not reveal a significant reduction of the average discovery time as the number of SDNCs increases from 1 to 5. This result is expected since generated topologies have a small network diameter, in terms of delay. Moreover, this behaviour remains similar across topologies with different connectivity degrees. Consequently, the proposed discovery mechanism can be considered irrespective of the network topology.

The average number of packets handled per switch required to discover the abstract network view is shown in Fig. 3(b). As expected, this metric is related with the average network degree. Moreover, in these topologies the required number of packets increases along with the number of SDNCs until levelling off at around 3-4 SDNCs. Thus, it may be inferred that the SHTD protocol is scalable with respect to the number of SDNCs, a characteristic that is crucial in practical applications.

Additionally, we show in Fig. 3(c) the reduction in the average number of packets managed by SDNCs considering two existing approaches as baseline, namely OpenFlow Discovery Protocol (OFDP) [3] and OFDPv2 [4], derived from equation $(NumPkt_{baseline} - NumPkt_{SHTD})/NumPkt_{baseline}$. As no study shows the OFDP operation for several SDNCs, we assume the switch distribution among SDNCs obtained from SHTD protocol. In contrast with [3], [4], in SHTD protocol each SDNC has to send and receive only one packet from each of its active interfaces which decrease the burden on SDNCs.

To assess the performance of the proposed recovery mechanism, in Fig. 3(d) we consider multi-links failure assumption for each pair of network links in NobelGermany-like topologies. To get a sense of the recovery times achieved, we also include in this analysis a Rerun approach. This approach denotes the discovery mechanism once the switches spontaneously notified the failure to SDNCs. As expected, in all cases the recovery mechanism outperforms the Rerun approach in terms of required time to update the global network view at SDNCs. Furthermore, it is confirmed that for all generated topologies, the recovery mechanism needs less than 50 ms, exhibiting the compliance with carried-grade networks requirement. In addition, this behaviour is not influenced by the increase of SDNCs, showing the good scalability of this proposal. It is worth to highlight that, similar results have been obtained for the other two sets of topologies, but due to space limitation we decide to show the behaviour on NobelGermany-like topologies, as they present the higher network delay diameter.

How much control intelligence should remain in SDN switches is still an issue of ongoing debate [4]. Although our solution embraces the network control decoupling from forwarding devices, we envisage an autonomic SDN environment where services like TD and fault recovery are mainly supported by the data plane.

IV. Conclusion

In this letter, we have proposed a novel SHTD protocol for SDN-managed networks. It is designed to optimize the current TD service and its reliability in the face of network failures. To achieve this, it combines a layer 2 topology discovery mechanism with the self-healing attribute to enhance the control plane robustness. Simulation results show the time-efficiency and scalability of the proposed solution on key network metrics. Moreover, the recovery of control connectivity is assured within the 50 ms required for carrier-grade networks. As future work, we will extend this protocol considering current network state data in order to optimize recovered control trees.

References

[1] S. Khan, A. Gani, A. A. Wahab, M. Guizani, and M. K. Khan, "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," *IEEE Commun. Surv. Tut.*, vol. 19, no. 1, pp. 303–324, 2017.

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] Open Networking Foundation, "OpenFlow Switch Specification v1.5.1," April, 2015. [Online]. Available: https://www.opennetworking.org/.

[4] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in openflow-based software defined networks," *Comput. Commun.*, vol. 77, pp. 52–61, 2016.

[5] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle, "A survey of autonomic network architectures and evaluation criteria," *IEEE Commun. Surv. Tut.*, vol. 14, no. 2, pp. 464–490, 2012.

[6] Y. Jiménez, C. Cervelló-Pastor, and A. García, "Dynamic resource discovery protocol for software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 5, pp. 743–746, 2015.

[7] L. Ochoa-Aday, C. Cervelló-Pastor, and A. Fernández-Fernández, "Discovering the network topology: An efficient approach for SDN," *Adv. Distrib. Comput. Artif. Intell. Journal*, vol. 5, no. 2, pp. 101–108, 2016.

[8] J. Sánchez, I. G. B. Yahia, and N. Crespi, "Self-modeling based diagnosis of software-defined networks," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, London, UK, Apr 2015, pp. 1–6.

[9] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0-Survivable Network Design library," *Networks*, vol. 55, no. 3, 2010.

[10] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an openflow architecture," in *Proc. 23rd Int. Teletraffic Congr. (ITC)*, USA, Sept 2011, pp. 1–7.

[11] N. L. M. v. Adrichem, B. J. v. Asten, and F. A. Kuipers, "Fast recovery in software-defined networks," in *Proc. 3rd Eur. Work. Softw. Defin. Networks*, Budapest, HU, Sept 2014, pp. 61–66.