# Towards a Resilient In-Band SDN Control Channel

Polina Goltsman
and Martina Zitterbart
Institute of Telematics
Karlsruhe Institute of Technology
Email: {goltsman,zitterbart}@kit.edu

Arthur Hecker
Huawei European Research Center
Email: artur.hecker@huawei.com

Roland Bless
Institute of Telematics
Karlsruhe Institute of Technology
Email: bless@kit.edu

***Abstract*—In-band SDN control simplifies SDN deployment but introduces new challenges as the network control commands are running on top of the data plane they are configuring. We argue to put responsibility of maintaining connectivity of the control channel on the switches. Based on this idea this paper presents a design of a distributed protocol which focuses on resilience and low protocol overhead.**

## I. Introduction

In *software-defined networks (SDN)* the network control plane is physically separated from the data plane. A (logically-) centralized SDN controller is deployed on a dedicated node(s) within the network. This raises the question: how the connectivity between the two planes is established and maintained. The SDN controller requires a communication channel to each SDN switch which we refer to as an *SDN control channel*. Two principle design options exist: an *out-of-band* or an *in-band* control channel. The former runs over a separate management network, while the latter shares network links with normal network traffic. Although not maintaining a separate management network offers several benefits [1], maintaining connectivity of the in-band control channel is challenging, as controller's changes to the data plane may affect the connectivity of the control channel and, thus, the ability of the controller to manage the switches.

Connectivity of the control channel is critical for a seamless operation of an SDN-based network. In addition to physical failures, connectivity of the control channel can also be broken by (erroneous) management commands. Note, that the control logic is implemented in applications hosted on the controller. Such an application could, for example, issue a command to close a port, that is currently used by the control channel, thus introducing a link failure. If the connectivity of the control channel is maintained by the controller itself, as it is done in [1], [2], special mechanisms to prevent such situations are required. Contrary to these works, we argue that the control channel will be more resilient if the responsibility of maintaining the connectivity to the controller lies on switches and not on the controller.

We are currently developing the distributed protocol IS-C (intermediate system to controller) for maintaining connectivity of the in-band control channel. To keep the protocol overhead low, IS-C is tailored to the traffic pattern of the SDN control channel. Furthermore, it incorporates several resiliency mechanisms, including fast reroute [3], to minimize packet losses in case of failures. Within this short paper we present the basic concepts behind IS-C.

## II. Overview of IS-C

The design of IS-C is based on two main ideas. First, on the SDN control channel the traffic is flowing only between the controller and each switch and the protocol only needs to maintain these routes. Existing protocols, such as STP, OSPF or IS-IS/Trill, allow to route traffic between all pairs of nodes. Instead IS-C should reduce protocol overhead by calculating only the necessary paths. Second, the SDN control channel is critical for the operation of the network. Thus IS-C should aim at maintaining connectivity during recovery from failures.

IS-C is distributed, therefore, each switch hosts an entity of IS-C's control plane. The control plane entity communicates with its switch's data plane using OpenFlow. In particular, it installs rules in the OpenFlow pipeline. IS-C uses the IPv6 packet format, but assigns different semantics to individual fields.

Additionally, we defined following requirements for the protocol. As the space in OpenFlow tables is a limited resource, the protocol should minimize the necessary number of entries in the flow tables. The protocol should not require manual configuration. However, we assume that the switches have unique identifiers from a possibly flat address space.

The resulting protocol is consists of the following three main building blocks: (1) a spanning tree, (2) a labeling scheme [4], and (3) a fast reroute mechanism [3]. Below we describe the functionality of IS-C's control plane.

### A. IS-C in a Nutshell

The protocol calculates a minimum depth spanning tree rooted at the controller. The traffic is flowing only along the branches of the tree in both directions. We utilize a prefix-based labeling scheme to assign temporary addresses to nodes (see Fig. 1). The address of each node encodes its position in the tree. This allows prefix-based forwarding in the data plane if each node installs rules to its children. Therefore, IS-C keeps forwarding tables small, that is $\mathcal{O}(node\ degree)$, without requiring manually assigned hierarchical addresses. Finally, each node pre-calculates backup recovery paths that route around parent link and node. All of the sub-tasks are performed using distributed algorithms.

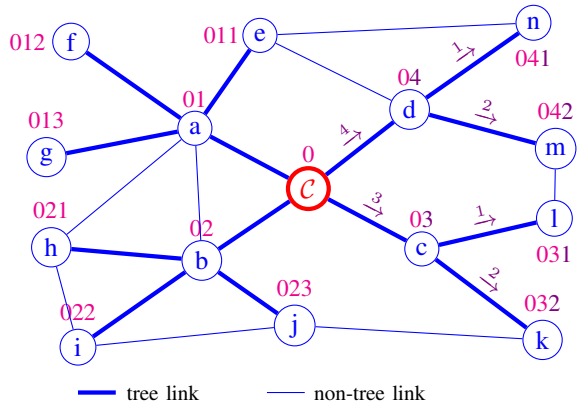


Figure 1. An illustration of the labeling scheme. Each node numbers its children (only shown for the nodes on the right) and this number is appended to the parent label. Each number is encoded as a fixed-width binary.

### B. Fast Reroute

Besides normal *primary routes* the protocol calculates alternative *recovery paths* that are used if links or nodes on the primary paths fail. We use pre-calculated recovery paths as this feature is supported by OpenFlow. The following scheme protects against a single link or node failure.

Each node calculates a recovery path to its tree parent (to protect against link to parent failure) and the parent of its parent (parent node failure). To find these paths nodes use breadth-first-search to learn the topology of their local neighborhood. We use a scheme that is similar to [5]. Once the paths are determined, the node uses signaling mechanism to install the routes on the nodes along the path. An example of a backup path is shown in Fig. 2.

### C. Handling Topology Changes

We aim to not disrupt packet flow during route recalculations. In IS-C the primary technique to accomplish this is to introduce sequence numbers to labels. The labels have the form $\{$*sequence number, prefix*$\}$. When the tree is recalculated, the sequence number is increased and the prefixes are updated. This allows two versions of the tree to coexist temporarily: when the labels change, new rules are added to the data plane and an expiration timeout is set on the old rules. The packets can be forwarded using the "old" tree while the "new" tree is being calculated.

With fast reroute employed, the protocol does not need to recalculate the primary routes immediately after failures. Instead we use periodic convergence with a fairly large period of several minutes to suppress convergence on transient failures. During the convergence, that is recalculating of the new tree and re-labeling, the nodes use old addresses and old routes. Once the convergence is finished and new rules are written to the data plane, the controller signals the nodes to use the new addresses. This ensures update consistency. The protocol utilizes distributed termination detection which is a known problem in the area of distributed systems. Since the old tree remains routable for this time, the packet flow is only disrupted by failures not covered by fast reroute.

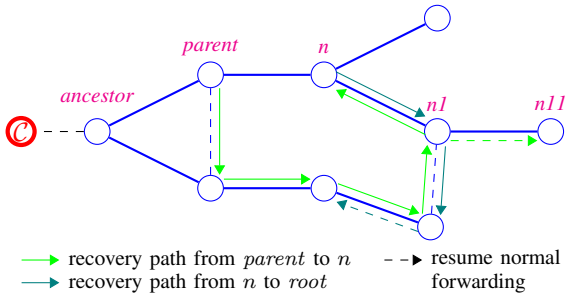


Figure 2. An example of recovery paths calculated by node $n$.

### D. Outlook

We are currently working on improving the following two features of the protocol. First, the fast-reroute scheme will be extended to protect against several failures with at least high probability. Second, currently, each change in the topology requires re-labeling of the complete tree. Our goal is to replace this mechanism with a mechanism which requires relabeling of only the affected sub-tree.

## III. Related Work

Most current deployments use *out-of-band control* [2]. Centralized protocols to maintain connectivity of the in-band control channel are developed, for example, in [1], [2]. In both projects, the routes for the control channel are installed and maintained by the controller. Similar to our work, [6] proposes that the switches, not the controller, maintain the control channel connectivity. Their work is complementary to ours as [6] focuses on the integration of a distributed protocol for the control channel into an SDN network while using an existing protocol OSPF as a proof-of-concept. We focus on the design of the protocol itself, including especially improved and built-in resiliency.

## References

[1] S. Sharma, D. Staessens, *et al.*, "Automatic bootstrapping of openflow networks", in *19th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*, IEEE, 2013.

[2] M. Canini, I. Salem, *et al.*, "A self-organizing distributed and in-band SDN control plane", in *37th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[3] M. Shand and S. Bryant, "IP fast reroute framework", RFC 5714.

[4] N. Santoro and R. Khatib, "Labelling and implicit routing in networks", *The computer journal*, vol. 28, no. 1, 1985.

[5] W. Tavernier, D. Papadimitriou, *et al.*, "Self-configuring loop-free alternates with high link failure coverage", *Telecommunication Systems*, vol. 56, no. 1, 2014.

[6] T. Omizo, T. Watanabe, *et al.*, "Resilientflow: Deployments of distributed control channel maintenance modules to recover sdn from unexpected failures", *IEICE Transactions on Communications*, vol. 99, no. 5, 2016.