Chapter 2.4 Reliable Control and Data Planes for Softwarized Networks

C. Mas Machuca, F. Musumeci, P. Vizarreta, D. Pezaros, S. Jouët, M. Tornatore, A. Hmaity, M.Liyanage, A. Gurtov, A. Braeken

Abstract Software Defined Networking is a new paradigm that extracts the inherently distributed control plane of forwarding network elements such as switches and routers, to a logically centralized control entity referred as SDN controller. The SDN controller acts as a broker between the network applications (e.g., monitoring, traffic engineering) and the data plane (i.e., physical network infrastructure). For scalability and robustness, the logically centralized control plane is implemented by physically distributing different controllers throughout the network. This chapter presents different solutions to increase the reliability of the data plane and the control plane of SDN networks. The reliability of the data plane can be increased by considering survivable virtual network embedding solutions as well as taking advantage of a programmable data plane. The reliability of the control plane can be addressed by considering enhanced controller placement solutions providing redundancy against uncorrelated as well as targeted failures while coping with latency and capacity re-

C. Mas Machuca, P. Vizarreta
Technical University of Munich (Germany), e-mail: cmas@tum.de, petra.stojsavljevic@tum.de

F. Musumeci, A. Hmaity
Politecnico de Milano (Italy), e-mail: francesco.musumeci@polimi.it, ali.hmaity@polimi.it

D. Pezaros
University of Glasgow (UK), e-mail: dimitrios.pezaros@glasgow.ac.uk

S. Jouët
VeryConnect (UK), e-mail: simon@veryconnect.com

M. Tornatore
Politecnico di Milano (Italy) and Univ. of California (USA), e-mail: massimo.tornatore@polimi.it

M.Liyanage
Univ. College Dublin (Ireland) and Univ. of Oulu (Finland), e-mail: madhusanka.liyanage@oulu.fi

A. Gurtov
Linkping University (Sweden), e-mail: gurotv@acm.org

A. Braeken
Vrije Universiteit Brussel (Belgium), e-mail: abraeken@vub.be

quirements. Furthermore, a solution to increase the security and robustness of the control channel is also addressed in this chapter.

1 Introduction

Software Defined Networking (SDN) is a recent paradigm that aims at increasing network flexibility and efficiency by separating the control from the data plane. The data plane consists of interconnected forwarding devices (e.g., switches, routers) that forward packets based on their forwarding tables. These tables are built by the control plane, which is the intelligent layer. The control plane configures the paths at the data plane based on the requirements from the application layer and also provides an abstract view of the data plane to the application layer. Data flows can be set up based on new flows from connected users or on the requests from the application layer. In the former case, the forwarding device will contact the controller to know how to proceed. Despite of the logically centralized approach of the control place, it can be physically distributed at different locations. In that case, forwarding devices (e.g., switches) are assigned to at least one controller (more than one for resilience purposes). Coordination among the controllers is required (e.g., following federation or hierarchical architectures).

Data plane resilience can be approached from different fronts. The first approach deals with the protection and restoration of data flows. Existing protection schemes for transport networks such as dedicated or shared path protection, which finds link and/or node disjoint paths can be also applied to SDN networks. These schemes aim at offering 100% reliability and have been further extended in order to consider QoS/security aspects and use less resources when possible. The first proposed techniques are e.g., by Xie et al. [51] proposing a proactive local failure recovery module running at the forwarding components able to restore flows in case of one local failure. Furthermore, each controller implementation offers different restoration and/or protection approaches to address failure scenarios, which can be further extended (e.g., the POX controller offers several algorithms extended by Vaghani et al. [48]). Section 2.1 presents a Survivable Virtual Network Embedding to ensure content and network connectivity. Last but not least, Section 2.2 introduces the concept of programmable data planes to achieve resilience.

The control plane in SDN is logically centralized, but for scalability and reliability purposes, it is implemented in several controllers. Any forwarding device of the data plane is connected at least to one controller (referred as primary controller). The loss of connectivity between the forwarding devices and their designated controllers, as well as the failures of the controllers themselves, will seriously diminish the overall network performance. Section 3.1 introduces existing concepts and alternatives to increase the control plane reliability. Section 3.2 proposes a solution able to cope with disaster scenarios so that the SDN increases its robustness. Last but not least, Section 3.3 addresses the security limitations of the control channel.

2 Reliable Data plane

2.1 Survivable Virtual Network Embedding for Content-Connectivity

When deploying network resources, telecommunication services providers usually aim at ensuring the so-called Network Connectivity (NC) against single-link failure, that is, they guarantee that all node pairs are still connected after any isolated failure in the network. However, if a disaster occurs, several links might be affected, thus maintaining service continuity during such catastrophic events is a non-trivial challenge. Providing NC after multiple physical links have been interrupted might require a very high amount of redundant network resources or might not even be possible due to the structure of the physical network. Moreover, after a disaster occurs, a significant amount of time may be necessary to perform network recovery and restore NC. Therefore, research efforts have recently been focused on a new concept called Content Connectivity (CC) [16], to ensure the reachability of the content from any point in the network even in case of multiple failures.

Fig. 1 shows a particular example of content connectivity where data-centers are located at nodes `A` and `D`. Note that if physical links `BC` and `FC` fail simultaneously, e.g., due to a large disaster; the network connectivity cannot be guaranteed, but the content connectivity can be still maintained as all nodes can reach one data-center.

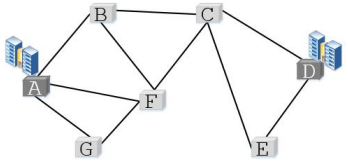


Fig. 1: Content connectivity vs network connectivity: if datacenters are located at nodes `A` and `D`, contents can be reached by any node even after network disconnection (e.g., due to failure of links `BC` and `FC`) [18].

2.1.1 Survivable Virtual Network Embedding

The problem of guaranteeing content connectivity against disasters (i.e., multiple failures) can be expressed as a Survivable Virtual Network Embedding (SVNE) problem, where the embedded virtual network is constituted by a set of data-centers[1] to be distributed across the physical nodes [18]. SVNE consists of assigning physical network resources to the virtual network requests such that the virtual network is survivable to failures in the physical network. Let us present an illustrative example of SVNE considering both NC and CC.

Example: Consider the virtual and the physical networks shown in Fig. 2a and Fig. 2b, respectively. In the virtual network, the connectivity is requested between

[1] Note that each of the data-centers is assumed to host all the requested contents.

node pairs AF, AE, FD and ED. Besides these, content-connectivity is requested by all the virtual nodes in the network, i.e., all nodes shall be connected to any of the two data-centers, assumed as located at nodes A and D. To perform the SVNE, all the links of the virtual network shall be mapped on a path at the physical topology.

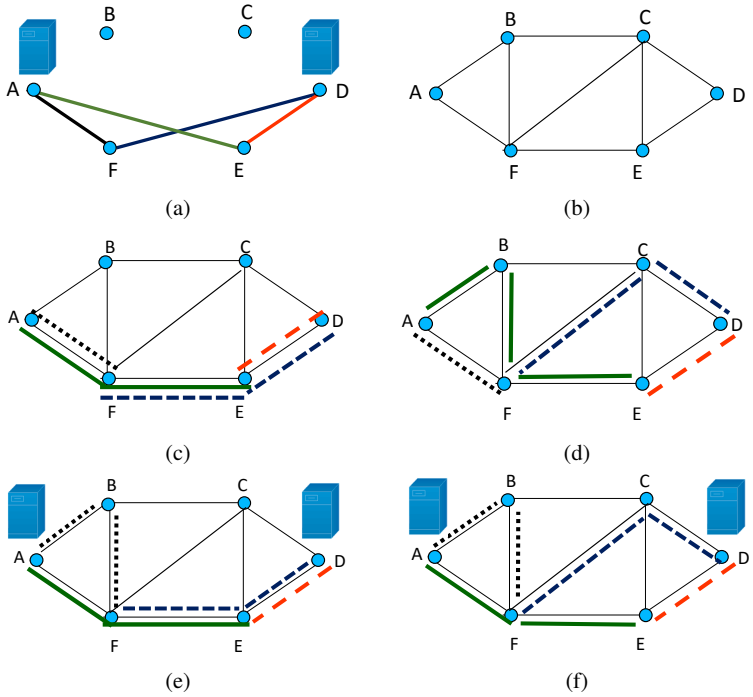


Fig. 2: Content-connected (CC) vs Network-connected (NC) in a Survivable Virtual Network Embedding (SVNE) problem: (a) Virtual network, (b) Physical network, (c) Non-survivable mapping, (d) CC mapping against single-link failures (CC1), (e) NC mapping against single-link failures (NC1), (f) CC mapping against double-link failures (NC1+CC2) [18].

Fig. 2c shows a non-survivable VNE, even against a single-link failure, as failure of physical link FE would produce the disruption of virtual links FD and AE, hence the virtual network would be disconnected. On the contrary, Fig. 2d shows a network-connected SVNE against single-link failures. In fact, any single link failure in the physical network would not cause disconnection of the virtual network. This, however, is obtained at the price of using extra network resources in the network.

More important, in the case of disasters, simultaneous failures of multiple physical links can occur, and this work refers to the case of survivability against double-link failures and focuses on content-connected embedding. Fig. 2e shows a content-connected survivable embedding against single-link failures (CC1), while Fig. 2f depicts an embedding which provides both NC against single-link failures (NC1)

and maintains CC against double-link failures (CC2). Such design leverages replication of contents in data-center sites coupled with the SVNE, and it is denoted with NC1+CC2. Note that in case physical link FE fails, the embedding shown in Fig. 2e can still provide the reachability to content in data-center locations, for all virtual nodes. Successively if physical link AF fails, the virtual network will be disconnected, as virtual node F cannot reach any data-center location. In case SVNE and content replication are combined within data-center locations, as shown in Fig. 2f, the embedding provides NC after single-link failures and virtual nodes A, F, E and D can still reach one data-center location in case of double-link failures.

2.1.2 Problem statement

The SVNE problem for network- and content-connectivity against double-link failures can be stated as follows.

Given the virtual and physical network topologies and the set of datacenter locations in the physical network, **decide** on a mapping of each virtual link onto a physical path **such that** after any double-link failure the virtual network remains connected and each node in the virtual network can reach at least one data-center location. As a typical **optimization objective**, the minimum amount of physical network resources, e.g., physical links capacity, can be pursued.

For a more formal problem formulation, the reader is referred to [18], where the authors propose an ILP-based mathematical model considering an optical network as the underlying physical network, so that the objective is to minimize the number of wavelengths used in the network.

2.1.3 Case-study and results

In [18], the authors compare the NC and CC solutions against double-link failures with the corresponding cases where single-link failure resilience is guaranteed. Specifically, the authors define the following four scenarios:

- network connectivity against single-link failures (NC1);
- content connectivity against single-link failures (CC1);
- network connectivity against double-link failures (NC2);
- content connectivity against double-link failures (CC2).

Note that if network-connectivity is ensured after one (respectively, two) failure(s), content-connectivity is ensured by definition, whereas the vice-versa case does not necessarily hold. Therefore, an interesting issue arises with a VNE where network connectivity is guaranteed after any single-link failure, while content connectivity is ensured also after double-link failure. This case is referred to as NC1+CC2 (Figure 2f) and might represent a reasonable trade-off between required network capacity and service availability.

Several virtual topologies with increasing connectivity degree have been considered to perform numerical evaluation, taking the NSFNET topology as the physical

network. Moreover, the effect of the number of datacenters in the network has been also investigated. It has been shown [18] that for the considered case studies:

- in CC1 almost as many resources as the NC1 scenario are required for any logical topology, that is, the virtual topology has no significant impact on the resource usage in case of a single-link failure;
- in case of double link failures, NC1+CC2 greatly reduces the required network resources if compared to NC2 when the virtual network is 3-connected (i.e., each virtual node has at least three neighbor nodes); however, the difference between NC1+CC2 and NC2 is less relevant when the connectivity degree of the virtual topology increases;
- for the discussed topologies, a limited number of data-centers is sufficient to guarantee content connectivity against double-link failures with minimum network capacity.

2.2 Programmable Data Planes for Resilient Software-Defined Networks

2.2.1 Problem definition

Data planes have been designed to be reliable by heavily distributing the logic across all the nodes in the network. Assuming a large number of redundant nodes and inter-connections, the traffic can be routed through different paths in case of node failure. This approach has been suitable to survive from critical failures, assuming a large enough number of backup routes. Providing high reliability using an over-provisioning of resources is costly as a large part of the infrastructure will be underutilised but it also fails to address new emerging challenges. Using a highly distributed system provides some reliability by preventing a single point of failure in the network but also prevents from making infrastructure wide decisions. Actual network attacks can be initiated by a large number of machines distributed across many different network providers [45]. This distribution results in an ingress of anomalous traffic that can remain undetected in the network using the node-local information available. Using infrastructure-wide information from many nodes of the infrastructure, global decisions can be made that can detect and react to these network anomalies.

SDN has emerged over the last decade as a paradigm to centralise the network's control plane and separate it from the underlying data plane. This physical separation gives the ability to program the control plane software, and it therefore allows the flexible development of services to, among others, tackle anomalies programmatically over large network infrastructures. However, the current *de facto* realisation of SDN through Openflow's match-commit framework provides insufficient programmability and only supports an inherently limited set of fields and actions [35]. Native services such as measurement-based resource provisioning and anomaly detection cannot be seamlessly supported by the current paradigm due to

their stringent timing and load requirements that would put excessive strain to the data-control plane interface.

In order to support such performance-bound services, the network fabric would need to also offer the ability to programmatically express the data plane behaviour of each network node. This would include the switch-local forwarding behaviour through protocol parsing and table lookups, but would also encode additional, more complex tasks such as, e.g., load balancing, anomaly detection, encryption and protocol offloading. Such tasks are currently not possible in OpenFlow, due to their stringent time requirements but also due to the restricted nature of the OpenFlow's current match-commit mode of operation. BPFabric [23] has been recently proposed, which is a protocol, platform, and language-independent SDN environment that facilitates both control and dataplane programmability and, through this, enables the development of arbitrary, high-performance data plane functions. It relies on a constrained High Level Language (HLL) for defining lightweight functions to perform a wide range of services such as statistics gathering and reporting, packet tracing, network telemetry, load balancing and anomaly detection. These arbitrary functions are subsequently compiled to the extended Berkeley Packet Filter (eBPF) instruction set, which serves as an intermediate interpretation to then allow for diverse hardware or software targets to be used while capturing the real-time constraints inherent to such networking devices. In Fig. 3 an overview of a programmable network is shown. In this network the centralized controller deploys network function within the data plane to achieve a target objective. For instance by deploying Intrusion Detection Systems (IDS) at the edges of the network anomalous traffic can be prevented and using Telemetry (TE) modules the state of the network can be monitored over varying timescales.

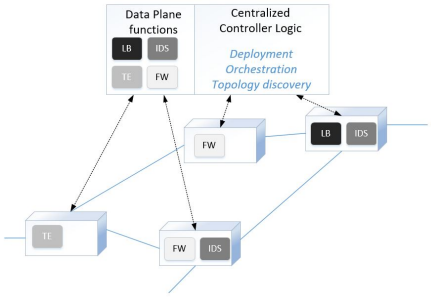


Fig. 3: Overview of a programmable and resilient data plane architecture with data plane functions: Intrusion Detection System (IDS), Load Balancing (LB), Telemetry (TE) and forwarding (FW).

2.2.2 Programmable Data Plane

Fig. 4 illustrates the communication between the controller and a single switch in BPFabric. The controller and the agents communicate through an API similar to Openflow's control plane protocol, to allow platform and protocol-independent programs to be deployed, events to be generated, and the state of each switch to be queried and updated. The logical flow for creating distinct data plane behaviour in the proposed architecture is shown in Figure 4:

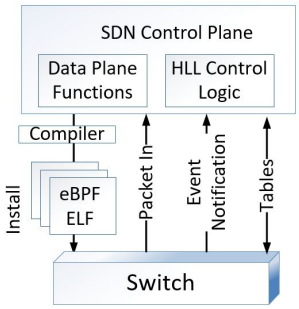


Fig. 4: BPFabric controller-to-switch architecture

- **Data plane behaviour:** It is defined using a HLL which dictates the parsing, matching, and processing that is performed on every packet.
- **Behaviour compilation:** The constrained HLL definition is compiled to a platform and protocol-independent instruction set.
- **Controller Install Request:** A control plane program can deploy the data plane functions to the required switches in the network.
- **Agent Function Loader:** The agent running on the switch receives the function form the controller and processes it into a suitable format for the target device (hardware or software). This includes the allocation of the necessary match tables to construct a packet processing pipeline.
- **On Ingress Packet:** Every ingress packet passes through the processing pipeline, a sequence of table query and update operations, while notifications can also be raised to the controller, if necessary. Finally, a forwarding decision is made.
- **Forwarding decision:** Based on the output of the pipeline, the packet can be forwarded to the relevant output port, sent to the controller, dropped or flooded to all (other) switch ports.
- **Controller operation:** If notified by the switch(es), the controller can be delegated the responsibility of deciding the forwarding decision for a packet.

Using the above approach, the controller can deploy new user-defined functions into the data plane, ranging from traditional routing and forwarding functions to more complex middlebox-like functions that can be used for constructing resilient services. In order to be able to deploy data plane functions across heterogeneous

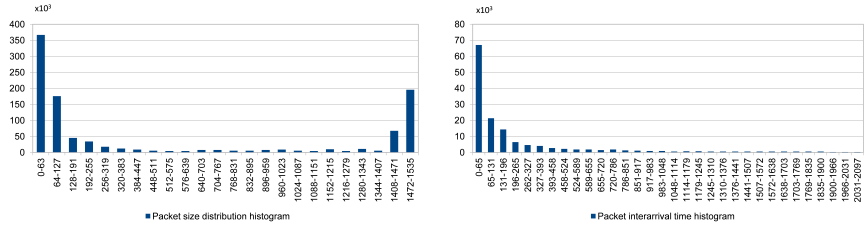
network node architectures, the compiled function needs to be platform independent. For this purpose, the extended eBPF instruction set has been exploited, as the intermediate representation for the compiled data plane functions. eBPF provides protocol independence, allows the parsing of arbitrary protocols, and supports the definition of a set of tables to maintain state and perform match-action operations.

The proposed architecture lets users specify how and where the packet headers should be parsed, preventing today's issues with OpenFlow requiring the specifications to be extended continuously to match the demand for additional processing. In addition, platform independence allows the same compiled program to be deployed across a large range of devices without requiring knowledge about the device, and without requiring each program to be recompiled for every new type of device.

2.2.3 Case study and results

Network Telemetry

Always-on network telemetry is used for verifying the network's normal behaviour and for identifying anomalies as deviations from such behaviour. Enabling network resilience typically requires granular visibility into changes occurring in the network, and subsequent reaction in short timescales. Telemetry data can therefore be used to verify that the network is behaving as intended, and to monitor the changes that occur over time. Through the always-on collection of carefully defined network metrics, the network controller or a third-party management application can gain granular visibility into the network behaviour in order to model and predict future trends. This insight can be used to adapt the fabric as the demand changes, migrate applications or virtual machines (VMs) to improve reliability [12], as well as improve policies to meet with the customer's Service Level Agreements (SLA) [10, 11]. Moreover, telemetry data can be exploited in synergy with other logs to identify potential anomalies, security threats, and also for debugging [9, 22].



(a) Packet size distribution histogram (b) Packet interarrival time histogram

Fig. 5: Telemetry histograms derived from dedicated data plane functions

Telemetry support has traditionally been offered by network devices through different implementations providing different levels of insight into the network traffic. For example, the Simple Network Management Protocol (SNMP) includes a widely

standardised and adopted implementation of telemetry under both pull and push models. Vendor-specific values can be periodically pulled from the devices by a management station, while network devices can also asynchronously report events of interest using the trap mechanism. Using BPFabric, two indicative telemetry examples have been implemented, which use the pull and push (devices stream events of interest directly to data collection points) approaches, respectively, demonstrating how user-defined telemetry metrics are defined and collected from the infrastructure. Fig. 5 shows two screenshots of the controller real-time visualisation of the reported metrics by the switches:

- **Packet Size Distribution**: This is a data plane function that stores a histogram of the packet size distribution of packets entering a switch over time (Fig. 5a). The controller subsequently queries/pulls the current state of the histogram in any switch implementing this function using "the table list" control message. This module can be used to characterise the evolving nature of traffic and its corresponding applications. Identifying the evolving trends in traffic over a long period of time can be used to profile network normal behaviour and therefore detect potential anomalies when the traffic deviates from normal.
- **Packet Inter-Arrival Time**: The second telemetry example measures the packet inter-arrival time distribution at a particular switch (Fig. 5b). This is achieved through the use of two tables, to store the histogram of inter-arrival times, and to keep track of the time of the last packet received, respectively. The histogram data is subsequently pushed to the controller at regular intervals. Packet inter-arrival times, and the associated mean and jitter can provide congestion-state indicators for the network, information about the traffic burstiness that impacts queuing delays and real-time streams, and also help with traffic classification.

Lightweight Anomaly Detection

This example demonstrates how functionality with stringent timing requirements can be integrated with the main forwarding operation as part of a programmable data plane. Such functionality that has traditionally been delegated to middleboxes can be implemented natively as through lightweight middlebox-like packet processing pipelines.

A lightweight anomaly detection algorithm is implemented using Exponential Weighted Moving Average (EWMA), a statistic that averages data over time giving more significance (weight) to recent measurements than ageing ones. Using this statistical value provides insight on the normal operating condition of the network while capturing the evolving behaviour of traffic. Significant deviation from this normal behaviour can then highlight anomalies within the network. Multiple network metrics can be used to represent the network behaviour, such as the number of packets, their size, inter arrival time or the volume of traffic transmitted or received. In our example, the EWMA value of the volume of traffic received on every port of the switch is computed. At a specific time interval (e.g., 5s), the EWMA value is calculated and compared against the threshold values. If the computed value exceeds the threshold, a notification is raised to the controller signalling an anomaly.

Fig. 6 shows the reported and predicted EWMA values and shows a (constructed) anomaly taking place at $t = 600s$.

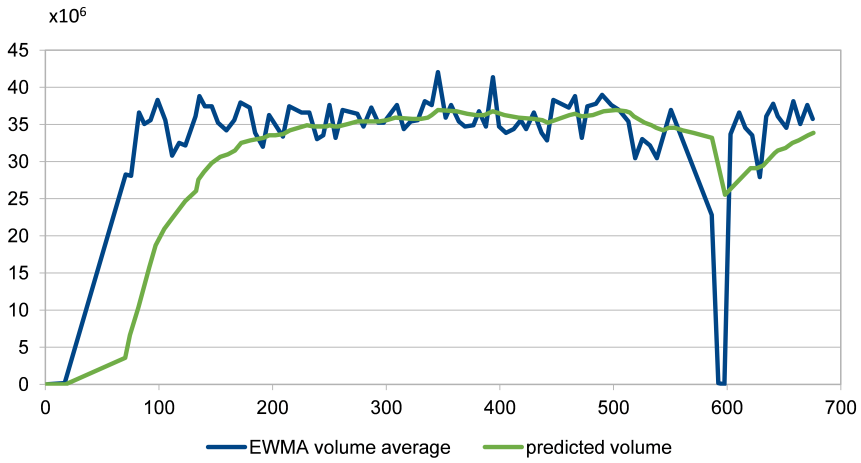


Fig. 6: EWMA volume average in 10^7 bytes (blue) and predicted volume (green) over time (0-700s). An anomaly (link failure) is introduced at t=600s

The above examples illustrate how lightweight network functions can be introduced natively as part of the main forwarding operation of the datapath at each switch, enabling programmable functionality and hence facilitating the development of services with stringent time requirements (unsuitable for being implemented purely at the control plane). Such services can include, among others, adaptive resource provisioning, infrastructure protection, and network resilience.

3 Reliable Control plane

3.1 Resilient Controller Placement Strategies

The Controller Placement Problem (CPP) deals with the decision on how many controllers are required and where should be placed for a given network so that all nodes have an assigned controller. The first problem definition was proposed in [17] to find the placement of k controllers (i.e., $|S| = k$) at node locations (i.e., $s \in V$). The problem has been considered for different metrics defined for a graph $G(V,E)$:

- average case latency [17]: Finds the placement of k controllers so that the average latency from any node to its controller is minimized. In this case, the average latency L_{avg} is defined as:

$$L_{avg} = \frac{1}{n} \sum_{v \in V} \min_{s \in S} d(v,s) \tag{1}$$

where $d(v,s)$ is the latency between the node $v \in V$ to its controller s;

- worst-case latency [17, 21]: The placement aims to minimize the latency of the furthest node to its assigned controller. In this case, the maximum node to controller latency L_{wc} is defined as:

$$L_{wc} = \max_{v \in V} \min_{s \in S} d(v,s) \tag{2}$$

- maximum cover [17]: In this case, given a maximum allowed latency between controller and assigned nodes, the controllers have associated a set of nodes they can reach within this latency. The k controllers are selected so that the size of the union of their associated sets is maximized;
- inter-controller latency [19]: In order to improve the synchronization between controllers so that they are updated with the network state, the placement of the controllers should either minimize the inter-controller latency L^{cc}_{avg} or to minimize the worst case inter-controller latency L^{cc}_{wc} defined as:

$$L^{cc}_{avg} = \frac{1}{n} \sum_{s' \in S} \min_{s \in S} d(s',s) \tag{3}$$

$$L^{cc}_{wc} = \max_{s' \in S} \min_{s \in S} d(s',s) \tag{4}$$

- load balancing [19, 47, 52]: Given the controller processing capacity and the flow setup rate of the nodes, the number of nodes that can be associated to a controller is limited. In order to avoid congestion of some controllers and to minimize the communication time between node and controller, load distribution may be balanced among controllers.

The first solutions aimed at minimizing single metrics: e.g., latency (average and worst case) from switches to their assigned controllers [17]. Further work considered more metrics simultaneously and propose solutions to find the Pareto optimal solutions [4, 19, 25]. Let us focus on how reliability has been considered in the CPP.

3.1.1 Problem definition

The reliability of the CPP addresses the placement of controllers taking into account failures, attacks and/or disaster scenarios. These scenarios differ on the type and number of under-performing components (e.g., links, nodes, controller). Furthermore, it has to be considered that the control plane includes not only the controller itself, but also the communication between the controller and its associated nodes (so-called control flows) and the inter-controller communication.

Depending on the origin of the faults, they can be classified as intentional or unintentional faults. The former are referred as failures and the latter as attacks. Disasters are defined as a large set of correlated failures such as the ones caused by floods or earthquakes [33]. In most of the existing work, the impact of each type of component's fault is:

- link: when a link fails, all the data and control flows mapped to it, are interrupted. It is a hardware fault (e.g., a cable cut);
- node: a faulty node implies that all data and control flows going though that node are interrupted. It can be caused by hardware or software faults (e.g., power cut, bug);
- controller: when a controller fails, the existing data flows keep working but no new flows can be set or existing flows teared down. Controller failures may be caused by hardware (e.g., hardware malfunction on the machine hosting it) or by software [49, 50].

  The original SDN architecture considers a logically-centralized control plane based on physically distributed architecture. However, the control plane can be implemented with distributed controllers for reliability and scalability purposes. The difference between a logically centralized and a logically distributed architecture based on physically distributed architecture is as follows:

  - logically distributed architecture (so-called split architecture): controllers differ on their responsibilities inside the network, on the view of the network (each controller has a partial view of the domain of the network it is responsible for) as shown in Fig. 7(a);
  - logically centralized architecture consists of different controller replicas. In this case, all controllers have the same responsibilities and they split the charge equally. However, hard synchronization is required so that all controllers are aware about any change in the network. This option is depicted in Fig. 7(b). One particular application is for 1:1 protection, as shown in Fig. 7(b), as one controller is the primary controller, and in case of its failure, the second controller (C1 is the back-up controller) becomes the active controller.

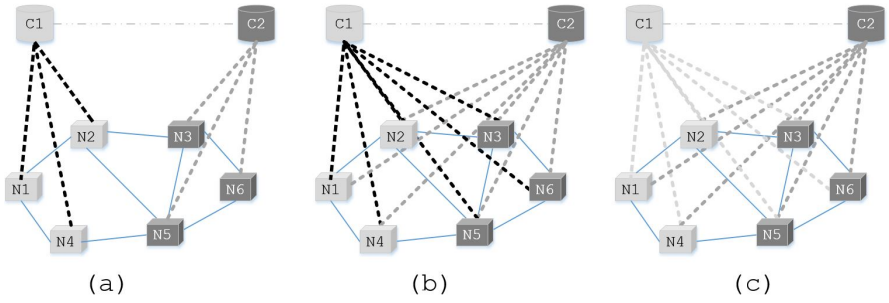


Fig. 7: (a) Logically distributed architecture (b) Logically centralized architecture (c) 1:1 protected architecture

It has been shown that the controller location impacts significantly the robustness of the control plane [53, 19]. Hence, this section summarizes several CPP solutions aiming at improving the reliability of the control plane.

3.1.2 Single/Multiple fault scenarios

Let us present existing solutions addressing single and/or multiple fault scenarios.

- min-cut clustering [53]: This work considers the logically distributed architecture shown in Fig. 7(a). The work proposes a CPP using a min-cut based graph partitioning algorithm to increase the resilience between controllers and nodes. The work considers single and independent faults at the nodes and links (i.e., excluding controller faults);
- node connectivity [38]: This work approaches the CPP problem considering two metrics: network disruptions and controller overhead. The approach consists of two parts: first, it places the controllers in order to maximize the number of node-disjoint paths between each node and its assigned controller, while keeping any given capacity constraints. Then, it computes an ordered list of back-up controllers in case the assigned one cannot be reached. The study evaluates *resilience equation* [53] (probability of node-controller connectivity in a uniform failure scenario), *cardinal of edge-connectivity* [6] (probability of node-controller connectivity considering all failure scenarios), *number of overloaded controllers* [38] for all failure scenarios and *load distribution* [38]. This work considers independent single and multiple failures of any network component.
- minimum control flow reliability: The CPP proposed in [42] places the controllers such that the availability of the control flow of any network node is higher than a given threshold. This work considers independent failures of links, nodes and controllers, which are given by failure probabilities. The results show that regardless of the network topology, each node has to connect to two controllers on average to guarantee five nines availability.
- control flow percentage [20]: This paper proposes a CPP to minimize the Expected Control Path Loss (*ECPL*) metric. *ECPL* is computed for every failure scenario and evaluates the percentage of lost control flows. Let us denote as F_s the set of all failure scenarios, $p(f)$ the probability that failure scenario $f \in F_s$ occurs and $C(f)$ the percentage of lost control flows when f occurs. Then,

$$ECPL = \sum_{f \in F_s} p(f)C(f) \tag{5}$$

  The evaluation compares the performance of a greedy algorithm, the proposed simulated annealing approach and the brute force solution. The results in [20] show that the ECPL is minimized when the number of controllers is slightly lower than 10% of the nodes.
- Protected control flows [49]: This work proposes increasing the control plane reliability with two approaches:

- disjoint Control Paths (RCP-DCP): there are two disjoint paths between every node and its assigned controller. This approach copes with single failures in the data plane but not with controller failures;
- different Controller Replicas (RCP-DCR): every node must be connected to two different controllers over two disjoint paths. This approach copes with single failures including controller failures.

In order to evaluate the impact of these proposed solutions, the control path length (related to the switch to controller delay), the *ECPL* and the average control path availability A_C are evaluated. A_C averages the availability of all the control flows from any node i in the network, as expressed by:

$$A_C = \frac{1}{|V|} \sum_{i \in V} A(i) \tag{6}$$

The comparison of the two proposed approaches versus the unprotected scenario presented in [49] shows a significant increase of the control plane resilience with a limited penalty added to the control path length. In order to cover link and node failure scenarios, RCP-DCR outperforms RCP-DCP.

3.1.3 Disaster and attack scenarios

The CPP has also been addressed by considering multiple failures scenarios such as disaster scenarios with several link and/or node failures (e.g., [43, 44]). The worst case scenario is referred as malicious attack scenarios which aim at causing the failure of the links and/or nodes that are more critical, that is, whose failure affects the network at most. More information on this robust CPP problem is presented in Chapter *Structural methods to improve network robustness to attacks* in detail.

3.2 Disaster-resilient control plane design

Transport Software-Defined Network (T-SDN) is expected to become a fundamental part of telecommunication infrastructure to provide efficient services for future technologies (5G and beyond). In order to support such services, transport network will need to be adapted to different applications with different requirements (e.g., some applications may require high reliability at low latencies, while others may require very high bandwidth), host network functions, etc. Currently, most SDN solutions (developed for Layer 3) cannot be applied to T-SDN without proper adaptations; a T-SDN control plane design must address heterogeneity in terms of protocols and administrative network areas, as well as high reliability. Hence, a robust hierarchical control plane is crucial in T-SDN.

3.2.1 Problem Definition

Transport networks pose different characteristics compared to other network domains (e.g., L3) with the implemented SDN paradigm. The following considerations are most critical in the control-plane design for T-SDNs [5, 46]:

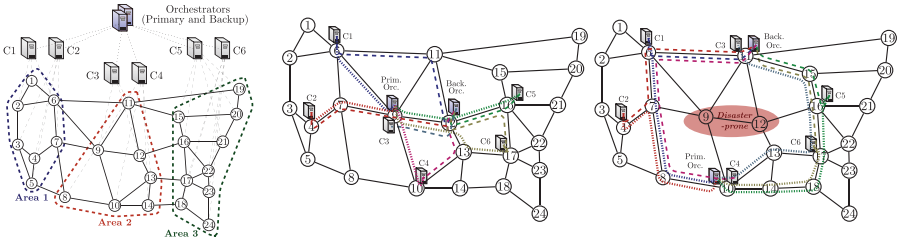
- *heterogeneity:* transport networks support diverse co-existing technologies and architectures (such as SONET/ SDH, OTN, etc.) which may be common to geographical regions of the network, and/or to services with varying bandwidth, latency, or availability requirements. Manufacturers impose proprietary technologies and specialized management systems, which makes several *areas*[2] in the network administratively isolated from each other and unmanageable by a single common entity.
- *high-Reliability:* in transport networks, a link or node failure can be very damaging, as they commonly connect several densely-populated areas and/or high-bandwidth infrastructures (e.g., Data Centers).
  Single or even double element failure preparedness may not be sufficient (some *Service Level Agreements* may require five nines of availability or higher). Hence, it is important to ensure survivability against even unlikely events such as natural or human-made disasters [37, 39].

To deal with above two factors, T-SDN focus on a hierarchical control plane [32]. The lower level of this hierarchical structure is composed of controllers, each capable of managing elements in a certain network area. Controllers, then, communicate with a centralized entity, named orchestrator, so that the network may provide services across different areas. Such controllers and orchestrators can either be physical machines or virtualized systems.

A method to design a robust, hierarchical control plane for T-SDNs is proposed in [32] such that it provides redundancy against uncorrelated failures, as well as disaster survivability, while considering latency and capacity constraints among the switches, controllers, and orchestrators. To deal with heterogeneity, the solution assigns switches of different network areas to the respective controllers, decides how many and where to place controllers in each area, and connects them to a central, primary orchestrator (and also backup orchestrators), in a hierarchical architecture. It also decides how to route switch-to-controller and controller-to-orchestrator traffic, avoiding shortest paths when more reliable, longer paths are also within latency bounds. To provide high reliability, the solution: i) provides redundancy of control-plane elements for failure resiliency (i.e., multiple controllers, backup orchestrators, and paths connecting them); ii) efficiently places controllers and orchestrators in the network (while choosing what paths to connect them), to minimize the effects of disasters. The method provides all of the above while respecting other system-related requirements (e.g., maximum control traffic latency or controller capacity).

The model assumes that the hierarchical control plane is structured as follows. First, each network element belongs to some *network area*. All elements of a certain *area* are managed (e.g., through the controllers' Southbound Interfaces [5, 46])

[2] In [5], *domain* is the equivalent of a *network area*.



(a) (b) (c)

Fig. 8: (a) Logical topology of a hierarchical control plane of a T-SDN with three *network areas*. Each area has two controllers and all controllers communicate with primary and backup orchestrators simultaneously. Dashed black lines represent controller's communication with switches. Dotted black lines represent controller's interactions with orchestrators. (b) A possible disaster-unaware control-plane placement (note that switch-to-orchestrator traffic is omitted). Dotted paths connect controllers to the Primary Orchestrator. Dashed paths connect controllers to the Backup Orchestrator. (c) A disaster-aware control-plane placement.

by one or more controllers specific to elements of that *area*. Controllers are then assigned to a centralized orchestrator (e.g., through their Northbound Interfaces [5, 46]) which allows for different areas of the network to operate together. Fig. 8a shows the logical structure of a hierarchical control plane.

Given the premise of a hierarchical control plane which must be made as robust against failures as possible, the focus is on optimally deciding:

- where to place controllers and how to assign switches;
- where to place orchestrators; and
- how to route switch-to-controller and controller-to-orchestrator traffic.

In the example of Fig. 8a, a possible solution for the control-plane placement is shown in Fig. 8b. This placement is a disaster-unaware placement and chooses controller locations and paths such as to minimize resource utilization (in terms of a number of links used for control traffic). Note that such an approach tends to concentrate many control-plane elements towards the center of the network topology (i.e., close to nodes `9` and `12`). In Fig. 8c, the same control-plane hierarchy as that of Fig. 8a is placed in a disaster-aware manner. Since nodes `9` and `12` are within a disaster-prone region, these nodes should be avoided when placing control-plane elements and when routing control-plane traffic.

To make the control plane resilient to disasters, network vulnerability can be determined by assessing what elements might be affected by each disaster (e.g., a map of disaster-affected regions as shown in Fig. 9). A probabilistic metric called *disaster risk* [13] can be computed to encompass the probability of a disaster occurring at a given time; and, given the occurrence, the probability that it will cause damage to network elements in the affected region. For set of all disasters that may

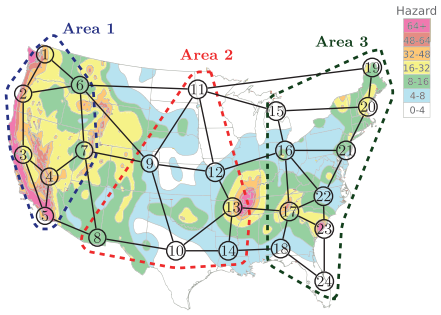


Fig. 9: A 24-node network topology over US seismic hazard map [41]. Colored regions represent possible disaster areas; their color-coded hazards represent *disaster risks*.

occur, a metric called *Disaster Impact* can be computed. This metric captures the expected loss of control-plane connectivity (i.e., links carrying switch-to-controller and controller-to-orchestrator traffic) due to disasters weighted by the risk of each disaster. It is defined as:

$$I = \sum_{y \in Y} \sum_{p \in P^c} R_y \cdot \left(1 - U_p^y\right) \tag{7}$$

where Y is the set of all disasters y, P^c is the set of all paths through which control traffic is carried, R_y is the risk (i.e., probability of occurrence) of disaster y, U_p^y is a binary variable, equal to 1 if path p is not affected by disaster y.

3.2.2 Problem Statement

Given: Network topology; which nodes are capable of hosting controllers and/or orchestrators; the association of network node to the network area it belongs to; which are the disaster-prone regions and their risks; maximum switch-to-controller and controller-to-orchestrator latencies (possibly different); how many switches each controller can manage (also called *controller capacity*); pre-calculated paths for each pair of nodes; number of orchestrators to be deployed in the network.
Output: How many controllers and where to place them; where to place orchestrator(s); what switches to assign to each controller; and how to route switch-to-controller, and controller-to-orchestrator connections.
Objective: First: Minimize the risk of control-plane disruption due to disasters. Second: Provide the solution with the minimal resource utilization (in terms of number of controllers and links utilized to route control traffic).

The above-mentioned objectives are realized through an Integer Linear Programming (ILP) model [32] which is to be used offline when deciding where to place the T-SDN hierarchical control plane. The objective of the formulation is stated as [32]:

$$minimize\,(\omega + \delta + \lambda) \tag{8}$$

where ω gives the average number of controller-to-orchestrator and switch-to-controller paths that fail due to disasters, weighted by the risk of each disaster occurring; δ gives the total number of nodes that host controllers (for all areas in the network) plus the average number of controllers that fail due to disasters, weighted by the risk of each disaster occurring (this term ensures that not only the least amount of controllers are deployed, but also that the deployed set of controllers is the one least impacted by disasters); and λ gives the resource utilization measured by the number of links used for switch-to-controller and controller-to-orchestrator communication. The first focus is on making the controller-to-orchestrator communication disaster-resilient; then, the switch-to-controller; and, finally, on first minimizing number of controllers and then the number of links used for control traffic.

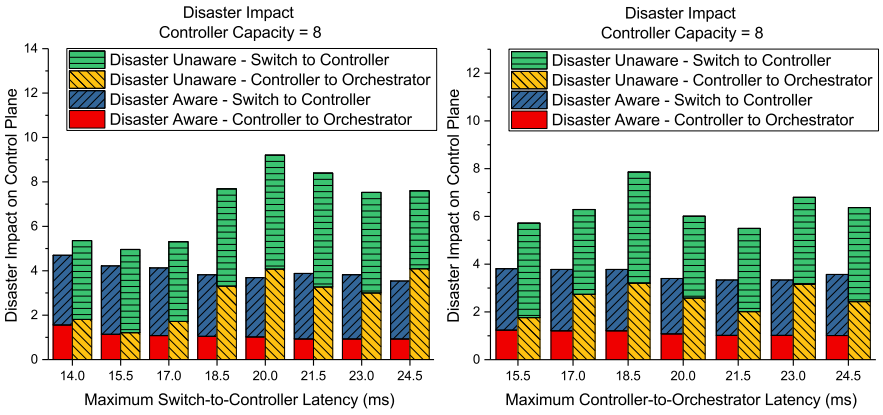
This method can be used to statically decide how to place hierarchical control plane in a robust manner. However, severe failures might still affect T-SDN control-plane operations. Once the placement of control plane is done, a heuristic [32] is developed for dynamic post-failure switch-controller reassignment to overcome such disruptions in either switch-to-controller or controller-to-orchestrator interactions.

3.2.3 Case-study and results

To evaluate the solution's resiliency against disasters, Fig. 9 is considered for topology and hazard map. In both disaster-aware and unaware approaches, it is enforced that every switch will have at least a second controller within reach, in case the first fails. As each area needs at least two controllers, the minimum number of controllers is six in the example, which has three areas. The results analyze how the disaster-aware and disaster-unaware solutions behave under different latency constraints, both for switch-to-controller and controller-to-orchestrator traffic.

Fig. 10 shows how the proposed *disaster-aware* solution compares to *disaster-unaware* solution with regards to the Disaster Impact *DI* metric. As an example, given two alternative control-plane designs for the network of Fig. 9 (with the same number of switch-to-controller and controller-to-orchestrator paths), a control plane whose $DI = 8$ has paths that traverse areas twice as risky as those of a control plane whose $DI = 4$, or traverse twice the amount of disaster-prone regions of similar risk. In Fig.10a, the maximum possible switch-to-controller latency is varied while not imposing limits to the controller-to-orchestrator latency. In Fig.10b, the opposite is applied.

The results show that the method provides a solution whose *DI* is much smaller than that of the disaster-unaware model, in all scenarios. Note, however, that the lengths of switch-to-controller and controller-to-orchestrator paths directly affect the placement solution. As these paths are allowed to be longer, the disaster-aware solution is able to find better paths, allowing the *DI* to decrease. The disaster-unaware solution does not have a clear relationship with this metric. Note that the red bars (i.e., controller-to-orchestrator paths in the disaster-aware model) are always smaller as the latency limits increase, which is not necessarily true for the



(a) (b)

Fig. 10: Comparison of the Disaster Impact (*DI*) of a disaster-unaware solution with the disaster-aware proposal. (a) shows how the maximum switch-to-controller latency affects the *DI* on the control plane. (b) shows how the maximum controller-to-orchestrator latency affects the *DI* on the control plane.

blue bars (i.e., the switch-to-controller paths). That is due to higher importance on the paths that connect controllers to orchestrators.

In Fig. 11, it is shown how the disaster-aware average control-plane path latency compares to the disaster-unaware average control-plane path latency. As the disaster-unaware model simply minimizes network-resource consumption, this solution will always find the shorter paths possible within the model's constraints. In Fig. 11a, the maximum possible switch-to-controller latency is varied while not imposing limits to the controller-to-orchestrator latency. In Fig. 11b, the opposite is applied. Fig. 11 shows how the disaster-aware solution's paths tend to be slightly longer than for the disaster-unaware solution. Note that the solution is more affected by constraints in the lengths of controller-to-orchestrator paths which causes it to compensate with longer switch-to-controller paths, as demonstrated in longer average switch-to-controller path latency (Fig. 11b) when compared to Fig. 11a.

As T-SDNs are heterogeneous and require high reliability, a robust hierarchical control plane is considered. Results from the illustrative examples where a transport network is susceptible to earthquake damage shows that the model achieves much higher disaster and failure resiliency with minimal extra resource consumption, when compared to a disaster-unaware approach.

3.3 Securing the control channel of SDNs

One of the most important security limitations of the SDN control channel is the lack of IP level security. Existing SDN control protocols rely on higher layer secure

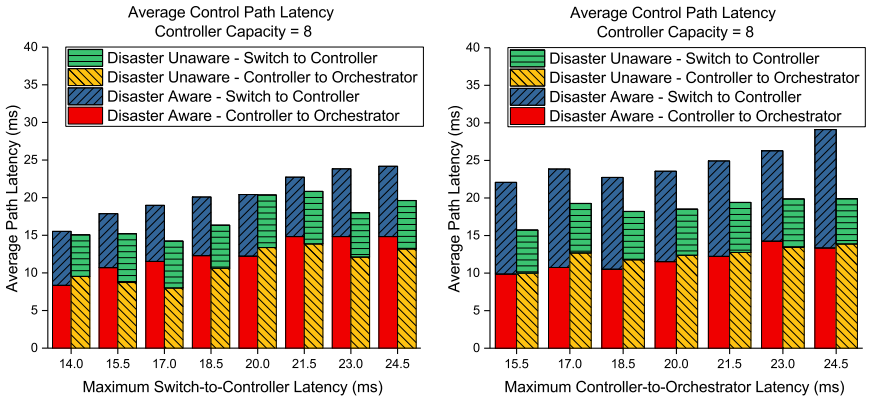


Fig. 11: Average control-plane latency: (a) shows how the maximum switch-to-controller latency affects average control-plane latency. (b) shows how the maximum controller-to-orchestrator latency affects average control-plane latency.

mechanism such as TLS (Transport Layer Security)/ SSL (Secure Sockets Layer) sessions (e.g., the widely used OpenFlow (OF) [34]). However, higher layer secure mechanisms are vulnerable to IP based attacks such as IP spoofing, TCP SYN Denial of Service (DoS) and TCP reset attacks [2, 3, 24, 36]. Furthermore, the security level of a TLS/SSL session depends on factors such as certificate authority, self-signed certificates, etc. The security of a TLS/SSL session is limited by the security level of the weakest link among those factors [36]. Moreover, a strong authentication mechanism is required between the controller and the switches [29, 31]. Otherwise, intruders can impersonate as legitimate switches and launch security attacks on the control channel. For instance, the attacker can inject fake flow requests to perform DoS attacks [24]. Unfortunately, the authentication procedure used at TLS/SSL sessions is not strong enough (as already mentioned, the TLS/SSL authentication mechanism is vulnerable to IP spoofing and Compression Ratio Info-leak Made Easy (CRIME) attacks [29, 36]). Therefore, these mechanisms are not sufficient enough to provide the required level of robustness and security for the control channel [27].

The network controller is the key component of the SDN network due to its centralized intelligence and controlling abilities. As mentioned in Section 3.1.1, the security of the control channel is a key factor to ensure the proper communication with the controller and its assigned switches [24]. For example, a DoS attack on an SDN controller was demonstrated in [14] in which an attacker continuously sent IP packets with random headers to the controller, causing the congestion of the controller (i.e., being unable to deploy flow rules in the switches).

TLSv1 based communication is optional in the latest OF specifications due to its complexity of configuration [1], as it requires to generate network site-specific

and corresponding signed device certificates with site-wide private keys for the controller and the switches [7]. Therefore, many SDN equipment vendors have skipped the support for TLS in their switches, which leaves the control channel vulnerable to security attacks. Let us propose a mechanism to secure the control channel.

3.3.1 Proposed Secure Communication Channel Architecture

This section presents a novel IPsec-based communication channel architecture to secure the control channels between switches and controller(s). The proposed architecture is presented in Fig. 12. It is a "bump-in-the-wire" security architecture based on the Host Identify Protocol (HIP) [28, 30]. HIP contains a strong PKI (Public-Key Infrastructure)-based authentication mechanism, which uses unique Host Identifiers [40]. The security requirements of SDNs have been accounted while designing this security architecture, which overcomes the limitation in IPsec tunneling mechanisms. It also provides the required security features such as confidentiality, integrity, availability, centralized controlling and visibility [8, 26].

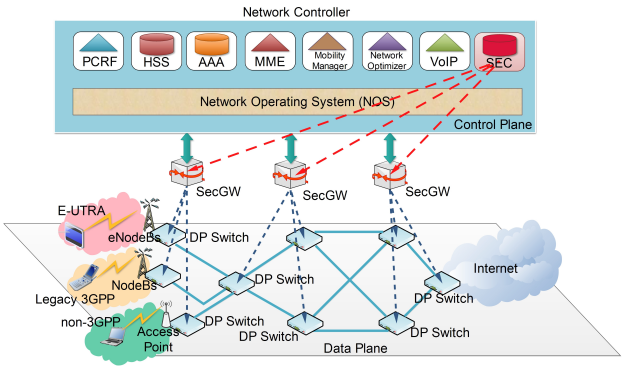


Fig. 12: Proposed IPsec-based Secure Control Channel Architecture

The HIP-based architecture proposes five main changes to the existing SDN architecture. First, distributed Security Gateways (SecGWs) are utilized to secure the controller from the outside network. Second, a new Security Entity (SecE) is added as a control entity to control the SecGWs and other security functions. Third, a Local Security Agent (LSA) is installed in each switch to handle security related functions. Fourth, IPsec Encapsulating Security Payload (ESP) Bounded-End-to-End-Tunnel (BEET) is used to secure the control and data channels communication. Fifth, session based Traffic Encryption Keys (TEKs) are used to encrypt the control and data channel traffic. Note that the introduction of the three entities, SecGWs, SecE, and LSA, offers a modular and easy plug-in solution for the establishment of the security features in the current infrastructure.

All the network control functionalities can be placed in a centralized location, enabling the creation of a trusted network zone for the important network elements in the SDNs. In this case, this central location hosts the administrative authority, responsible for the control of physical site locations, the ownership, and the operation of the network. Hence, the SecGWs are the interface between the trusted network zone and the outside world.

3.3.2 Performance Analysis of Control Channel

This section introduces the analysis of the performance penalty of security on the SDN control channel. Four laptops and two Ethernet hubs were used in the testbed. Two laptops with i5-3210M (2.5 GHz) CPUs were used as SDN switches. An Open-Vswitch (OVS) version 1.10.0 was installed in each laptop. Two virtual hosts (Host1 and Host2) were connected via OVS1 and they run Lubuntu 13.10 Operating System (OS). Similarly, two virtual hosts (Host3 and Host4) which run Lubuntu 13.10 OS, were connected via OVS2. The third laptop with a L2400 CPU of 1.66 GHz, worked as the SDN controller (latest POX). All three laptops had Ubuntu 12.04 LTS OS. They were connected via two D-LINK DSR-250N routers. The link speed of this experiment was set to 100 Mbps.

Connection Establishment Delay

The first experiment measures the connection establishment delay between the switches and the POX controller under different scenarios. This delay is measured based on ping requests from Host1 to Host2.The experiment results depicted in Fig. 13a show that the proposed architecture significantly increases (by almost 140%) the connection establishment delay. The HIP tunnel establishments between LSA and SecGW add extra delay. However, this delay increase can be minimized by keeping the established HIP tunnels for a long period. It has been shown that established HIP tunnels can be maintained for periods of 15 minutes [15].

Flow Table Update Delay

The second experiment measures the delay to update the flow tables for new packet flows in the steady state of operation, i.e., the HIP tunnels between LSAs and SecGW are already established and operational. This delay is measured based on ping requests from Host1 to Host2.The experiment results depicted in Fig. 13b reveal that the performance penalty of the proposed secure architecture is limited in the steady state of operation. The extra IPsec encryption increases the flow update delay only by 2%. However, this delay can be further minimized by using IPsec accelerators [28, 30].

4 Conclusion

The recently proposed Software Defined Networking (SDN) architectures allows separating the control and the data planes, which simplifies the required hardware components (to basic forwarding devices) combined with a logically centralized

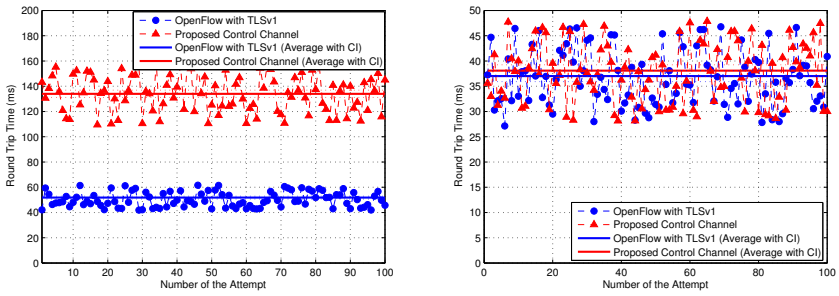


Fig. 13: Average control-plane latency: (a) connection establishment delay (b) flow table update delay.

control plane. This chapter addresses different solutions in order to increase the reliability of SDN solutions in both data and control planes.

References

1. OpenFlow Switch Specification Version 1.4.0.
2. I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtov. 5G security: Analysis of threats and solutions. In *2017 IEEE Conf. on Standards for Communications and Networking (CSCN)*, pages 193–199. IEEE, 2017.
3. I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtov. Overview of 5G security challenges and solutions. *IEEE Communications Standards Magazine*, 2(1):36–43, 2018.
4. V. Ahmadi, A. Jalili, S. M. Khorramizadeh, and M. Keshtgari. A hybrid NSGA-II for solving multiobjective controller placement in SDN. In *2015 2nd Intl. Conf. on Knowledge-Based Engineering and Innovation (KBEI)*, pages 663–669, Nov 2015.
5. R. Alvizu, G. Maier, N. Kukreja, A. Pattavina, R. Morro, A. Capello, and C. Cavazzoni. Comprehensive survey on t-sdn: Software-defined networking for transport networks. *IEEE Communications Surveys Tutorials*, 19(4):2232–2283, Fourthquarter 2017.
6. K. Bagga, L. Beineke, R. Pippert, and M. Lipman. A classification scheme for vulnerability and reliability parameters of graphs. *Mathematical and Computer Modelling*, 17(11):13–16, 1993.
7. K. Benton, L. J. Camp, and C. Small. Openflow vulnerability assessment. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 151–152. ACM, 2013.
8. J. Costa-Requena, J. L. Santos, V. F. Guasch, K. Ahokas, G. Premsankar, S. Luukkainen, I. Ahmed, M. Liyanage, M. Ylianttila, O. L. Prez, M. U. Itzazelaia, and E. M. de Oca. SDN and NFV Integration in Generalized Mobile Network Architecture. In *European Conf. on Networks and Communications (EUCNC)*, pages 1–5. IEEE, 2015.
9. L. Csikor and D. P. Pezaros. End-host driven troubleshooting architecture for software-defined networking. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conf.*, pages 1–7, Dec 2017.
10. L. Cui, R. Cziva, F. P. Tso, and D. P. Pezaros. Synergistic policy and virtual machine consolidation in cloud data centers. In *IEEE INFOCOM 2016 - The 35th Annual IEEE Intl. Conf. on Computer Communications*, pages 1–9, April 2016.

11. L. Cui, F. P. Tso, D. P. Pezaros, W. Jia, and W. Zhao. PLAN: Joint Policy- and Network-Aware VM Management for Cloud Data Centers. *IEEE Trans. on Parallel and Distributed Systems*, 28(4):1163–1175, April 2017.
12. R. Cziva, S. Jouët, D. Stapleton, F. P. Tso, and D. P. Pezaros. SDN-based virtual machine management for cloud data centers. *IEEE Trans. on Network and Service Management*, 13(2):212–225, June 2016.
13. F. Dikbiyik, M. Tornatore, and B. Mukherjee. Minimizing the risk from disaster failures in optical backbone networks. *Journal of Lightwave Technology*, 32(18):3175–3183, Sept 2014.
14. P. Fonseca, R. Bennesby, E. Mota, and A. Passito. A replication component for resilient OpenFlow-based networking. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 933–939. IEEE, 2012.
15. A. Gurtov. *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Wiley, 2008.
16. M. F. Habib, M. Tornatore, and B. Mukherjee. Fault-tolerant virtual network mapping to provide content connectivity in optical networks. *Optical Fiber Communication Conf. (OFC)*, pages 3–4, 2013.
17. B. Heller, R. Sherwood, and N. McKeown. The controller placement problem. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 7–12, New York, NY, USA, 2012. ACM.
18. A. Hmaity, F. Musumeci, and M. Tornatore. Survivable virtual network mapping to provide content connectivity against double-link failures. In *2016 12th Intl. Conf. on the Design of Reliable Communication Networks (DRCN)*, pages 160–166, Mar. 2016.
19. D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia. Pareto-optimal resilient controller placement in SDN-based core networks. In *Proceedings of the 2013 25th Intl. Teletraffic Congress (ITC)*, pages 1–9, Sept 2013.
20. Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan. Reliability-aware controller placement for software-defined networks. In *2013 IFIP/IEEE Intl. Symposium on Integrated Network Management (IM 2013)*, pages 672–675, May 2013.
21. Y. Jimenez, C. Cervello-Pastor, and A. J. Garcia. On the controller placement for designing a distributed SDN control layer. In *2014 IFIP Networking Conf.*, pages 1–9, June 2014.
22. S. Jouet, C. Perkins, and D. Pezaros. OTCP: SDN-managed congestion control for data center networks. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 171–179, April 2016.
23. S. Jouet and D. P. Pezaros. BPFabric: Data plane programmability for software defined networks. In *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 38–48, May 2017.
24. D. Kreutz, F. Ramos, and P. Verissimo. Towards secure and dependable software-defined networks. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 55–60. ACM, 2013.
25. S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann. Heuristic approaches to the controller placement problem in large scale SDN networks. *IEEE Trans. on Network and Service Management*, 12(1):4–17, March 2015.
26. M. Liyanage, A. B. Abro, M. Ylianttila, and A. Gurtov. Opportunities and Challenges of Software-Defined Mobile Networks in Network Security. *IEEE Security & Privacy*, 14(4):34–44, 2016.
27. M. Liyanage, I. Ahmad, A. B. Abro, A. Gurtov, and M. Ylianttila. *A Comprehensive Guide to 5G Security*. John Wiley & Sons, 2018.
28. M. Liyanage, I. Ahmed, J. Okwuibe, M. Ylianttila, H. Kabir, J. L. Santos, R. Kantola, O. L. Perez, M. U. Itzazelaia, and E. M. De Oca. Enhancing security of software defined mobile networks. *IEEE Access*, 5:9422–9438, 2017.
29. M. Liyanage, I. Ahmed, M. Ylianttila, J. L. Santos, R. Kantola, O. L. Perez, M. U. Itzazelaia, E. M. de Oca, A. Valtierra, and C. Jimenez. Security for future software defined mobile networks. In *2015 9th Intl. Conf. on Next Generation Mobile Applications, Services and Technologies*, pages 256–264. IEEE, 2015.

30. M. Liyanage, A. Braeken, A. D. Jurcut, M. Ylianttila, and A. Gurtov. Secure communication channel architecture for software defined mobile networks. *Computer Networks*, 114:32–50, 2017.
31. M. Liyanage, A. Gurtov, and M. Ylianttila. *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*. John Wiley & Sons, 2015.
32. R. B. R. Lourenço, S. S. Savas, M. Tornatore, and B. Mukherjee. Robust hierarchical control plane for transport software-defined networks. *Optical Switching and Networking*, 30:10–22, Nov. 2018.
33. C. Mas Machuca, S. Secci, P. Vizarreta, F. Kuipers, A. Gouglidis, D. Hutchison, S. Jouet, D. Pezaros, A. Elmokashfi, P. Heegaard, S. Ristov, and M. Gusev. Technology-related disasters: A survey towards disaster-resilient software defined networks. In *2016 8th Intl. Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 35–42, Sept 2016.
34. M. McBride, M. Cohn, S. Deshpande, , et al. SDN Security Considerations in the Data Center. *White Paper*, October 2013.
35. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM*, Mar. 2008.
36. C. Meyer and J. Schwenk. Lessons Learned From Previous SSL/TLS Attacks-A Brief Chronology Of Attacks And Weaknesses. *IACR Cryptology ePrint Archive*, 2013:49, 2013.
37. B. Mukherjee, M. F. Habib, and F. Dikbiyik. Network adaptability from disaster disruptions and cascading failures. *IEEE Communications Magazine*, 52(5):230–238, May 2014.
38. L. F. Mller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos. Survivor: An enhanced controller placement strategy for improving SDN survivability. In *2014 IEEE Global Communications Conf.*, pages 1909–1915, Dec 2014.
39. S. Neumayer, G. Zussman, R. Cohen, and E. Modiano. Assessing the vulnerability of the fiber infrastructure to disasters. *IEEE/ACM Trans. on Networking*, 19(6):1610–1623, Dec 2011.
40. J. Okwuibe, M. Liyanage, and M. Ylianttila. Performance analysis of open-source linux-based HIP implementations. In *2015 IEEE 10th Intl. Conf. on Industrial and Information Systems (ICIIS)*, pages 60–65. IEEE, 2015.
41. M. D. Petersen et al. Documentation for the 2008 update of the united states national seismic hazard maps. *US Geologic Hazards Science Center*, 2008.
42. F. J. Ros and P. M. Ruiz. Five nines of southbound reliability in software-defined networks. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 31–36, New York, NY, USA, 2014. ACM.
43. D. Santos, A. de Sousa, and C. Mas Machuca. Robust SDN controller placement to malicious node attacks. In *14th Intl. Conf. on the Design of Reliable Communication Networks (DRCN)*, pages 1–8, 2018.
44. D. Santos, A. de Sousa, and C. Mas Machuca. The controller placement problem for robust SDNs against malicious node attacks considering the control plane with and without split-brain. *Annals of Telecommunications*, Jul 2019.
45. S. Simpson, S. N. Shirazi, A. Marnerides, S. Jouet, D. Pezaros, and D. Hutchison. An Inter-Domain Collaboration Scheme to Remedy DDoS Attacks in Computer Networks. *IEEE Trans. on Network and Service Management*, 15(3):879–893, Sept 2018.
46. A. Thyagaturu, A. Mercian, M. McGarry, M. Reisslein, and W. Kellerer. Software defined optical networks (SDONs): a comprehensive survey. *IEEE Communications Surveys and Tutorials*, 18(4):27382786, 2016.
47. M. T. I. ul Huque, G. Jourjon, and V. Gramoli. Revisiting the controller placement problem. In *2015 IEEE 40th Conf. on Local Computer Networks (LCN)*, pages 450–453, Oct 2015.
48. R. Vaghani and C.-H. Lung. A comparison of data forwarding schemes for network resiliency in software defined networking. *Procedia Computer Science*, 34:680 – 685, 2014. The 9th Intl. Conf. on Future Networks and Communications (FNC'14)/The 11th Intl. Conf. on Mobile Systems and Pervasive Computing (MobiSPC'14)/Affiliated Workshops.
49. P. Vizarreta, P. Heegaard, B. Helvik, W. Kellerer, and C. Mas Machuca. Characterization of failure dynamics in SDN controllers. In *2017 9th Intl. Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 1–7, Sept 2017.

50. P. Vizarreta, K. Trivedi, B. Helvik, P. Heegaard, W. Kellerer, and C. Mas Machuca. An empirical study of software reliability in SDN controllers. In *2017 13th Intl. Conf. on Network and Service Management (CNSM)*, pages 1–9, Nov 2017.
51. A. Xie, X. Wang, W. Wang, and S. Lu. Designing a disaster-resilient network with software defined networking. In *2014 IEEE 22nd Intl. Symposium of Quality of Service (IWQoS)*, pages 135–140, May 2014.
52. G. Yao, J. Bi, Y. Li, and L. Guo. On the capacitated controller placement problem in software defined networks. *IEEE Communications Letters*, 18(8):1339–1342, Aug 2014.
53. Y. Zhang, N. Beheshti, and M. Tatipamula. On resilience of split-architecture networks. In *2011 IEEE Global Telecommunications Conf. - GLOBECOM 2011*, pages 1–6, Dec 2011.