GENEVIZ: a Visual Tool for the Construction and Blockchain-based Validation of SFC Packages

Muriel F. Franco[1], Martin J. J. Bucher[1], Eder J. Scheid[1], Lisandro Z. Granville[2], and Burkhard Stiller[1]

[1] Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
[2] Computer Networks Group, Institute of Informatics INF, Federal University of Rio Grande do Sul UFRGS
Av. Bento Gonçalves, 9500, Porto Alegre, Brazil
{`franco, scheid, stiller`}`@ifi.uzh.ch`, `martin.bucher2@uzh.ch`, `granville@inf.ufrgs.br`

Abstract. Network Functions Virtualization (NFV) decouples the network package performed by network functions from dedicated hardware appliance by running Virtual Network Functions (VNF) on commercial off-the-shelf hardware. Network operators can create customized network services by chaining multiple VNFs, defining a so-called Service Function Chaining (SFC). Because NFV became technically mature recently, the building of such SFCs still needs in-depth knowledge about NFV technology and its descriptors. Furthermore, there is a lack of tools that help to simplify the creation of SFCs. This paper, introduces *GENEVIZ*, a tool that provides a user-friendly interface for the creation of new SFCs as well as for importing and adjusting acquired SFCs (*e.g.*, from marketplaces of VNFs), in order to create new SFCs based on existing ones. Therefore, this work addresses as well data integrity and provides the functionality to store and validate SFCs through the use of blockchains. Three case studies are presented to provide evidence of the technical feasibility of the solution proposed.

Keywords: Network Functions Virtualization · Blockchain · Service Functions Chaining · Virtual Network Functions-as-a-Service.

1 Introduction

The paradigm of Network Functions Virtualization (NFV) has gathered significant attention over the last years both from academia and industry [11]. NFV decouples packet processing from dedicated hardware middleboxes and handles it within Virtual Network Functions (VNF) that run on off-the-shelf programmable hardware [10]. NFV offers several benefits, including simplified network operations, a potential of speeding up service delivery, and significant reductions in Operational Expenditures (OPEX) and Capital Expenditures (CAPEX) [8]. Also, NFV allows network operators to create customized network services by chaining together multiple VNFs (*e.g.*, firewalls, load balancers, and DHCP servers). Such network services can provide, for example, different levels of protection, performance, and connectivity for end-users, while the NFV-based virtualization reduces costs and increase the flexibility of the network (*e.g.*, accelerated time-to-market and dynamic resources allocation). Such aggregation of different VNFs building up a network service is represented as a Service Function Chaining (SFC).

As of today, a network operator must have in-depth knowledge about the NFV technology and its corresponding descriptors in order to create an SFC, which can be deployed on an NFV-enabled infrastructure to provide novel network services. In NFV an SFC is represented as a forwarding graph of VNFs and should take into account different descriptors representing the configurations and dependencies of each VNF that compound an SFC [5], such as the VNF Descriptor (VNFD) and the Network Service Descriptor (NSD). The task of dealing with each one of these descriptors is not trivial and requires efforts to configure each one of them manually. The process of constructing such an SFC is not intuitive, many manual steps are necessary for descriptors handling and editing, and the creation can be quite error-prone. This might even lead to a negative impact on a broader adoption of the NFV technology. Furthermore, by considering the prospective market growth of VNF-as-a-Service (VNFaaS) [2] and its potential to simplify the way how end-users obtain services in general, the lack of intuitive solutions has to be addressed when considering the potential of SFCs for end-users with no expertise in the NFV technology.

In this context, information visualization techniques are considered to be a viable tool to help network administrators understand the behavior of the managed network or service, in a faster and easier way [7]. Even though the analysis of such data can be almost fully automated, human interpretation plays a crucial role in decision-making process for network and service management. Especially in NFV environments an enormous amount of data is available and the understanding of it represents a challenging task itself [6]. Although past work exploited visualization techniques to simplify the identification of problems in SFCs, there is still a lack of research addressing the simplification of SFC construction. The visualizations can provide several benefits in NFV environments, such as *(a)* an intuitive way to select VNFs that will compound the forwarding graph and *(b)* a quick configuration of the SFC through its corresponding VNFs. Thus, the visual interface designed here also provides opportunities to reuse already available SFCs and check their integrity relying on the blockchains [1], making it easier to build a new SFC based on an existing one and also contributing to the expansion of the NFV business models (*e.g.*, marketplaces implementing VNFaaS approaches).

This paper introduces *GENEVIZ*, a visual solution allowing the construction of a new SFC Package based on multiple VNF Packages' information (*e.g.*, descriptors of each VNF) and other inputs defined through interactive visualizations (*e.g.*, minimum resources and dependencies to run the service). The tool proposed provides an easy way to create new SFCs as well as to configure properties of VNFs that will compound an SFC. Also, *GENEVIZ* is able to store the hash of the content of such an SFC package on a public blockchain to enable the verification of their integrity and origin (*i.e.*, the developer information) of an already existing SFC configuration before it will be deployed inside the network.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 introduces the *GENEVIZ*'s general architecture and prototype. An evaluation based on case studies is conducted together with a discussion in Section 4 in order to provide evidence of the effectiveness of *GENEVIZ*. Section 5 concludes this paper and outlines future work.

2 Related Work

The management of networks and services demands a multitude of methods, activities, procedures, and tools, with the goal to ensure a proper functioning of systems observed. Such tools enable a network administrator to retrieve management information from corresponding devices, analyze the obtained data, and take decisions to optimize or repair services. Within this workflow, visualizations can provide a way to represent a large amount of data in a way perceivable much faster by the human user than via raw and often abstract data. In such a direction, the information visualization allows to perform cognitive work more efficiently and hence in less time [3].

The field of information visualization applied to NFV environments [7] discusses the current applications of visualization and how it should be explored in different topics, such as NFV and Software-defined Networks (SDN). [15] examined specifically the process of configuring virtualized networks, making it clear that no tools existed to assist the configuration, deployment, and testing of virtualized networks by 2016. Specifically, they found no single graphical tool for the creation of a network map directly from a configuration as given by the various descriptors in an SFC configuration. [6] presented the *VISION* platform, which provides interactive and selective visualizations to assist NFV management. *VISION* not only helps network operators to identify and alleviate problems in the context of VNFs, but also provides a complete forwarding graph visualization. Although it provides useful information on incorrect VNF placements or performance problems using visualizations, it only focuses on services already deployed and monitoring systems previously configured, thus not addressing the creation of new network services through the help of visualization tools.

In the context of SFC visualization, [13] introduced SFCPerf, an automatic performance evaluation tool for SFCs. SFCPerf ensures the repeatability of the performance measurements by defining a testing workflow; thus, allowing the performance comparison among different SFC configurations based on the same test. The visualization module included in the tool provides a user-readable interface to visualize throughput, round-trip time, and request rate of a given SFC. Based on their scenarios, they discovered that the main impact factors on the overall performance of an SFC were *(i)* the number of physical link hops between different nodes, and *(ii)* the competition for resources on shared physical nodes. These visualizations can be useful, especially during the construction phase of an SFC, while considering different topologies and NFV platforms, ensuring that the performance meets the desired requirements. [4] presented the *SFC Path Tracer*, a troubleshooting tool for SFC environments that enables the visualization of the trace of network packets in SFC domains. This trace generation is accomplished by mirroring probe packets as they traverse through the chain. Hence, SFC Path Tracer can be useful for the identification of problems within an SFC configuration, as it pinpoints the origin of a possible problem by providing packet trace information. The authors also argue that the tool can be expanded in the near future to a more comprehensive measurement tool.

Although different solutions as discussed above address different aspects related to SFC (*e.g.*, placement, resources allocation, and performance), none of them is focusing on the simplification of the process of an SFC construction by providing intuitive tools (*e.g.*, based on information visualization) for end-users, who construct SFCs and configure their acquired VNFs before the start of the deployment.

3 *GENEVIZ*

This section introduces GENEVIZ (Generation, Validation, and Visualization of SFC Packages) and its conceptual architecture combined with relevant details of the prototype's implementation. *GENEVIZ* architecture is composed of separated, but interconnected components, providing flexibility to allow replacement of existing modules or adding new modules without affecting remaining components.

White blocks with solid borders of Figure 1 represent internal components and grey blocks with dashed borders represent external components (*i.e.*, decentralized). *GENEVIZ* is divided into three main layers: *User Layer*, *Data Layer*, and *Blockchain Layer*, respectively. Although the *Blockchain Layer* is not part of *GENEVIZ* itself, it is an integral part of the solution proposed, since it plays a crucial role for validate and trust in SFCs. *GENEVIZ (i)* simplifies the process of creating new network services (*i.e.*, SFCs) in general, and *(ii)* ensures data integrity of previously created services by validating them using blockchain.

An end-user accesses the tool through the *User Interface*, which provides interactive visualizations depending on data provided by the *Visualization Manager*. An integral part of this data is given by the *Template Catalog*, which retrieves templates from the *Templates Collector*. The *Collector* retrieves data from different sources (*e.g.*, marketplaces, independent catalogs, or a manual upload from the local machine). While creating an SFC through the *Service Constructor* visualization, VNF Templates from the *Template Catalog* are transmitted through the *Visualization Manager* to the *Management API* and stored through the *Package Handler* on the *Packages Database*. When an SFC Package is generated, the *SFC Package Generator* creates a package based on the information provided (*e.g.*, the list of included VNFs containing their descriptors, the

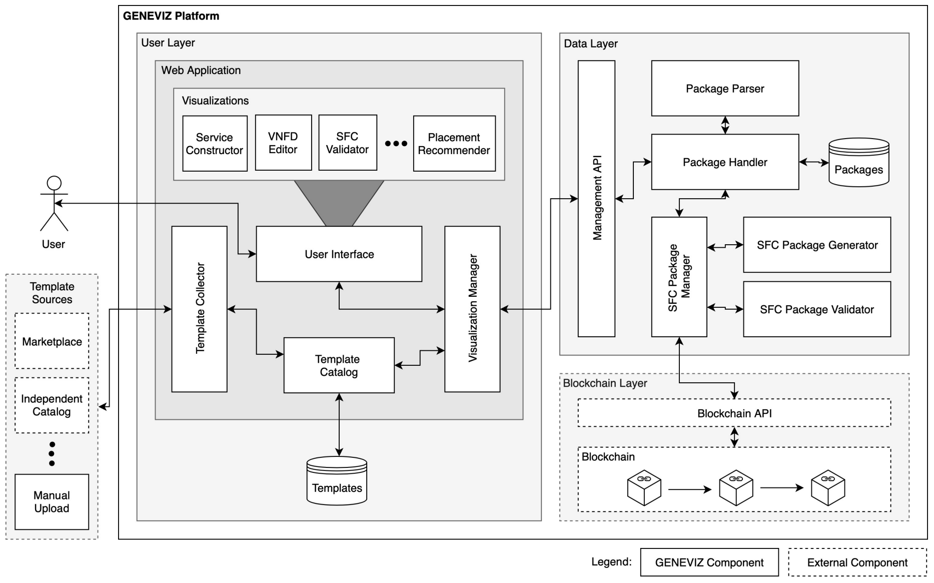


Fig. 1. GENEVIZ Architecture.

forwarding graph, and properties of the NSD). Besides, the end-user creating the package are able to select the option to store the hash of the SFC Package in the blockchain. If select and configure the blockchain information (*e.g.*, end-user's address and private key), the hash of such an SFC Package will be available on blockchain. After the transaction be sent to the blockchain, the transaction ID is stored in a descriptor called *geneviz.json*, which is included inside of the SFC Package, thus providing useful information to validate the hash during further validation by others.

The validation of the SFC Packages relies on the blockchain to verify the integrity of files representing the SFC Package. Blockchain was initially developed as a distributed ledger to be the backbone of the Bitcoin cryptocurrency [12]. Blockchain is an ordered list of blocks that uses cryptographic hashes to chain and identify the blocks. Each block has a dependency with the others on the chain, thus, if one wants to modify a data on the blockchain, he/she must change every block until the beginning of the chain. Because of the cryptography scheme implemented, this task is arduous and not viable in terms of computational resources. Based on that, blockchain ensures that one data stored cannot be removed, while the address of the account that stored the data and the data itself can be publicly available as well. Thus, among the benefits provided by blockchains, the trustworthy, decentralized, and immutable records can be highlighted as crucial for *GENEVIZ* to validate SFCs without the need to rely on any third-party (*e.g.*, marketplaces and public repositories for VNFs).

To enable the usage of the blockchain, *SFC Package Validator* communicates via the *SFC Package Manager* with the *Blockchain API*. The API retrieves the corresponding hash from the blockchain by using the transaction ID provided by the SFC Package. Theoretically, any blockchain (*e.g.*, Ethereum [17] and Bitcoin [12]) can be used for this validation. Only the hash of the SFC Package is required and can be stored, for example, using Smart Contracts or as a transaction in the blockchain. It also can be integrated with a blockchain-agnostic API, such as the one presented in [14]. Therefore, by using a public blockchain for the storage of the hashes of the SFC Packages, *GENEVIZ* allows for a check of the integrity and origin of that SFC Package. *GENEVIZ* implements Ethereum blockchain by default.

During the validation process, the *SFC Package Validator* compares the hash of the SFC Package, together with the given transaction key, with the hash stored on the blockchain for this transaction key, *GENEVIZ* can check if the package content matches the initial one from the creator of the SFC Package. Three possible states are defined: *(i)* Valid meaning that the hash of a SFC Package matches the one stored on the blockchain for the given transaction key, *(ii)* Invalid, when the SFC Package was modified and the hash of the downloaded package does not match with the one stored on the blockchain for the given transaction key; and *(iii)* Unknown represents that the SFC Package's hash was not stored in the blockchain during its creation because special reasons (*e.g.*, the developer decided to not use the blockchain validation upon constructing the SFC), thus, there is no transaction key or hash available for verification. In this case, *GENEVIZ* cannot make any statement about the integrity of the content (*cf.* Sec. 4.3 below).

3.1 Prototype and Implementation

The *GENEVIZ* prototype was implemented using JavaScript on the *User Layer* and Python 3.7.0 together with Flask 1.0.2 on the *Data Layer*. For the *Blockchain Layer*, Ethereum blockchain was used, supported by Ganache [16] its latest version as the development environment. For the SFC Package, the VNFD following the European Telecommunications Standards Institute (ETSI) standards and the codes to execute each VNF was considered. The prototype implemented considering those components previously defined, serving as a Proof-of-Concept (PoC) for the *GENEVIZ*'s architecture. The prototype's source-code and its documentation are publicly available online [9].

The left side of the *GENEVIZ* interface (*cf.* Figure 2) offers a menu, allowing the user to manually upload zipped VNF and SFC Packages by using the dropzone (depicted with the dashed border). By clicking on the respective buttons for *VNFs* and *SFCs*, the user can switch between these two catalogs. Only ZIP files containing VNF-related files are allowed to be uploaded. At the bottom of the menu, the blue button allows for the generation of an SFC Package based on the SFC constructed. This button only appears if the constructed SFC is valid. Hence, the button is not visible at the initial start of the application, as an empty SFC is considered to be invalid. The User Interface shows an alert on the top right corner if any error message has to be passed to the user. This can especially be helpful during the graph construction or import of new packages, since there is feedback from the application if an action fails or is not allowed. Examples include a wrong format of the VNF Template (*e.g.*, the *Package Parser* cannot find the VNFD) or the attempt to create a loop during the forwarding graph construction.

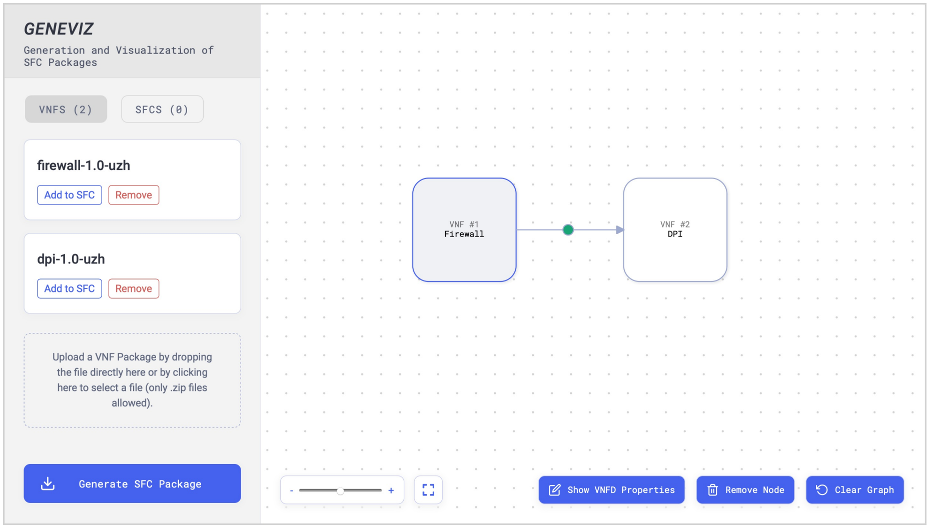


Fig. 2. GENEVIZ Dashboard.

GENEVIZ provides also feedback during the construction of the SFC (*i.e.*, chaining recommendation). When a new edge is being placed between two nodes on the graph plane, the edge color is defined based on the two properties *target_recommendation* and *target_caution* inside the VNFD of the source VNF. Such properties are an extension provided by *GENEVIZ* for the ETSI VNFD standard. These properties can be defined by the developer of the VNF inside of the respective VNFD. For example, dependencies of VNFs can be listed (*e.g.*, ensure that two complementary VNFs will be chained) and suggestions for performances improvement (*e.g.*, avoid the chaining of conflicting VNFs) can be described. Based on this information, *GENEVIZ* highlights wrong connections or possible bottlenecks during the SFC construction. Edges can be described as *(a)* green dot for a *recommended* target VNF, *(b)* red dot for *not recommended* target VNF, and *(c)* blue/white for neutral targets. The latter is especially important, since a VNF does not hold the entire list of all possible VNFs as targets in their properties, leaving the statement on the chaining recommendation neutral.

The structure of an SFC Package generated and handled by the *GENEVIZ* prototype is described as a *sfc.zip* file, which contains one or more VNF Packages. Each VNF composing the SFC has a separate folder for its respective content (*e.g.*, descriptors and source-code). For an SFC Package with multiple VNFs from the same template (*i.e.*, reuse of VNFs with different configurations), multiple VNFDs with different IDs will be available in the same *Descriptors* folder of the respective VNF. In addition, the *sfc.zip* file has a file named *nsd.json*, which represents the Network Service Descriptor (NSD) of the SFC.

Also, the *sfc.zip* file contains the *geneviz.json*, which is a *GENEVIZ* descriptor containing two properties, namely *txHash* and *address*. The transaction hash property *txHash* is retrieved from the blockchain itself after the hash is stored on the blockchain, and is known as the transaction ID of the transaction for the Ethereum blockchain. The *address* is given by the user itself when the generation of the package is requested. The *geneviz.json* is therefore needed for the validation of an SFC Package as it contains the transaction key necessary for the lookup of the data properties for this transaction key. If the data property retrieved from the transaction matches with the computed hash from the content of the *sfc.zip* file, the package is seen as *Valid*. Does the hash found on the blockchain for the given *txHash* not match with the computed hash from the content of the *sfc.zip* file, the package is seen as *Invalid*. If both the *txHash* and the *address* properties are empty strings, *GENEVIZ* has not stored any hash on the blockchain for this SFC Package and thus the data integrity of the SFC Package is presented as *Unknown*.

4 Evaluations and Discussion

In order to validate key features and the technical feasibility of *GENEVIZ*, three case studies on: *(i)* the process of the construction of SFCs, outlining benefits of the visualization to simplify the process of SFC construction, *(ii)* the generation of an SFC package, which means the merging of different VNF packages and configurations defined in previous steps. Also, the storage of the SFC package on the blockchain is shown, via an *(iii)* import of an existent SFC package (*e.g.*, available on online marketplaces or public repositories) and its validation of integrity and origin using the hash stored in the Ethereum blockchain. In addition, a discussion is provided to highlight the main benefits and limitations of the presented solution.

4.1 Case Study #1 - Construction of an SFC

Case study #1 considers a user with the specific demand to create a new network service, which shall be deployed in an NFV environment. Thus, the user bought three different VNF Packages from an external source (*e.g.*, from a marketplace), which are needed to create the SFC. The VNFs acquired implement a Deep Packet Inspection (DPI), a Firewall, and a Load Balancer (LB), respectively.

In a first step, the three VNFs are imported via the manual upload of the *GENEVIZ* Web application. After uploading them, all *VNF Packages* appear in the left menu as *VNF Templates*, since they could be added multiple times for the same SFC. By selecting the blue-bordered "Add to SFC" button for each *VNF Template* once, each template is added as a VNF Package to the SFC and appears as a node within the graph on the right side of the Web application. Next, the user constructs a connection between two VNFs selecting the DPI and Firewall. This leads to the creation of an edge between the DPI and the Firewall node. By connecting the Firewall with the LB node with the same approach, the second edge is created (*cf.* Figure 3).

This first draft of the SFC can be seen as a misconfiguration, although it would not be wrong to create such an SFC. The current construction also shows that the user is not experienced with the creation of SFCs, since a red dot on the edge between the DPI and the Firewall node appears. This red dot is part of the chaining recommendation of the *GENEVIZ*'s Prototype and indicates that the connection is not recommended for an SFC. In this case, this red dot highlights that a DPI allocated before a Firewall can generate a bottleneck in the service chaining. Hence, based on such an alert, the user changes the current construction and swaps the DPI and the Firewall node by deleting the two edges created, then swapping the position of the DPI and the Firewall node, and finally connecting the three nodes again by creating two new edges. As a result, the first edge will see a green dot, indicating a recommended connection. This recommendation is based on the information being part of the VNFDs, determining an additional as an extension of *GENEVIZ*.

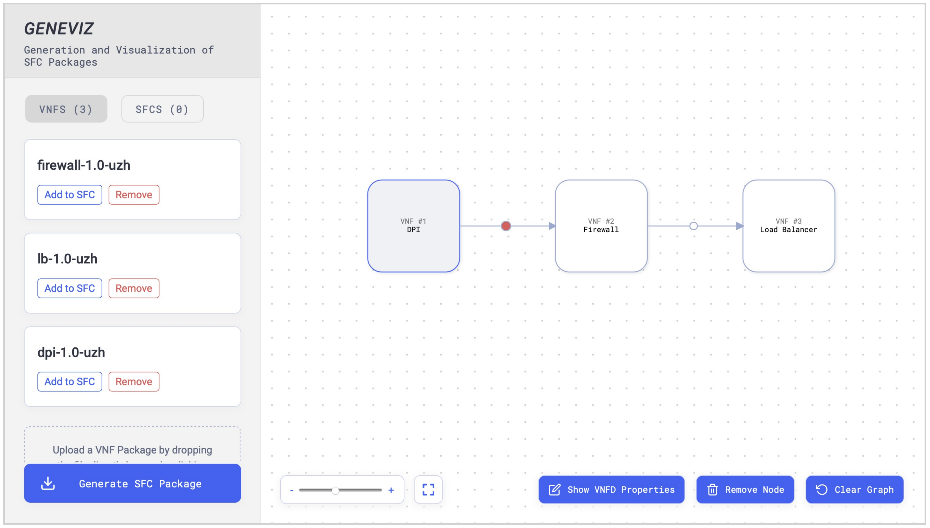
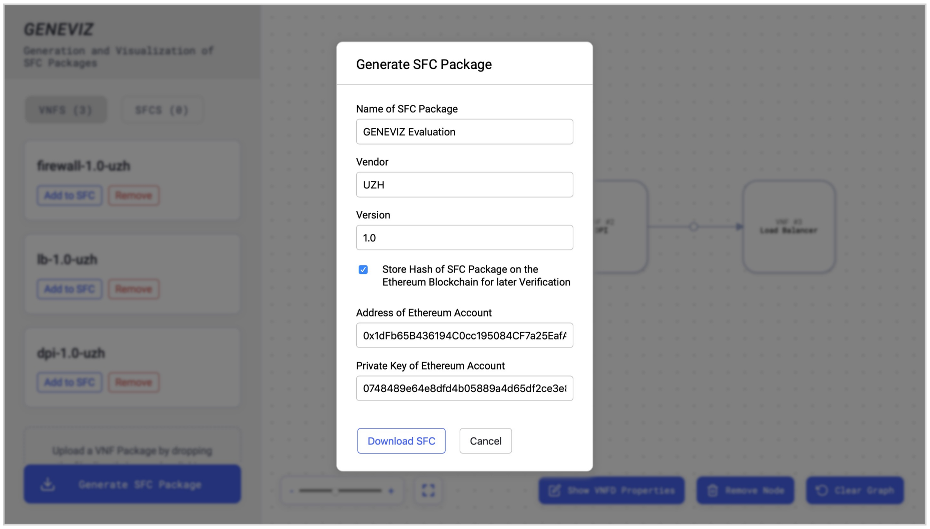


Fig. 3. SFC construction user interface for case study #1.

4.2 Case Study #2 - Generation of an SFC Package

The case study #2 evaluates the generation of an SFC Package and assumes a correct construction of the forwarding graph within the graph as constructed within case study #1. If such a constructed SFC is valid, a blue button with the label "Generate SFC Package" appears at the bottom left corner. By selecting on that a popup window appears, requesting the user to define the name, vendor, and version for SFC Package. If the end-user decided to store the hash of the SFC Package on the Ethereum blockchain for further validations, additional information are requested (*cf.* Figure 4). For this, the end-user needs to provide both the address of an Ethereum account as well as the private key for this account in order to sign the transaction properly.

As a next step, the user selects the blue-bordered "Download" button, which generates the SFC Package and stores the hash of the package on the Ethereum blockchain. The *GENEVIZ* will automatically trigger the download of a ZIP file, containing both an *sfc.zip* file for the deployment of the SFC as well as a *geneviz.json* file to be used for further validations of the SFC Package. The


Generate SFC Package

Name of SFC Package: GENEVIZ Evaluation

Vendor: UZH

Version: 1.0

[✓] Store Hash of SFC Package on the Ethereum Blockchain for later Verification

Address of Ethereum Account: 0x1dFb65B436194C0cc195084CF7a25EafA

Private Key of Ethereum Account: 0748489e64e8dfd4b05889a4d65df2ce3e

Fig. 4. SFC Package generation including its storage on the Ethereum blockchain.

4.3 Case Study #3 - Blockchain-based Validation and Import of an SFC Package

For case study #3 a different user is being considered. This second user has downloaded three SFC Packages before (*e.g.*, from a marketplace or a public catalog of VNFs), without any information on their integrity and origin (*i.e.*, developers). As these packages contain different folders and nested files representing each VNF that compose the service, for which a check of their content may be time-consuming, the user leaves the validation of these package up to *GENEVIZ*. Here the user refers to use the SFC Package named *sfc-package-evaluation*, but he also considers the *sfc-package-other* and the *sfc-package-other-2* packages in case the first one turns out to be invalid.

In the first step, the user selects the “SFCS” button in the menu on the left side to switch to the SFC section. This section allows for the manual upload of SFC Packages through the browser as it is performed for the upload of VNF Packages. The SFC Packages uploaded appear on the SFC list as *SFC Templates* in a similar way uploaded VNF Packages are handled as *VNF Templates*. The validation of the packages uploaded is triggered automatically and the response depends on the current block time of the blockchain. For the package named *sfc-package-other-2* no statement on its data integrity can be made, since no information for the retrieval on the blockchain is provided, hence, it is marked as *Unknown*. The second package *sfc-package-other* appears to be marked as *Invalid*, which means the content of this package was modified. The third package *sfc-package-evaluation* is the package created within case study #2. Since the hash of this package was stored in the Ethereum blockchain during the generation of the package by the first user and the transaction ID is part of the SFC Package downloaded, *GENEVIZ* finds a hash for the given transaction ID, which matches with the hash of the content from the SFC Package uploaded. Hence, *GENEVIZ* marks the SFC Package uploaded as *Valid*, and shows the green “Valid” label, as shown in Figure 5 within the white boxes of the *SFC Templates*.

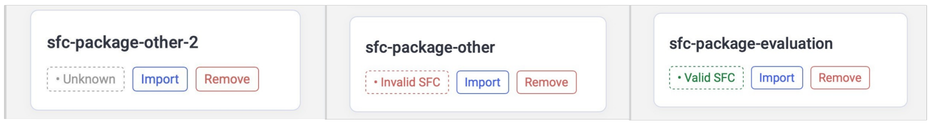


Fig. 5. Blockchain-based validation of SFCs.

For this third SFC Package, an acceptable level of trust was established, thus, the user decides to import this SFC Package as a new SFC into *GENEVIZ* in order to adjust VNFD properties of the VNFs involved, since they fit his/her demands. By selecting on the blue-bordered “Import” button on the left menu an alert appears, warning the user that the currently drafted SFC will be cleared and replaced by the new SFC. This is adequate for the user, since the drafted SFC previously is not relevant anymore. Now, the SFC will be imported and, if the SFC Package can be extracted, the visualization will be constructed to show the corresponding graph. This is considered helpful for the user, since the forwarding graph is now directly visible and can also be modified if needed. In turn, the user decides to open the *VNFD Editor* for the DPI, which lies between the Firewall and the LB. By selecting on the “Show VNFD Properties” at the bottom of the user interface (*cf.* Figure 3), a popup window appears, displaying current VNFD properties of the DPI. The user decides to change the memory size to “8 GB” and the number of CPUs to “2” to fulfill his/her demands for the new network service. After applying these changes, the new VNFD properties are directly updated. Finally, the user selects the option to generate the SFC Package, forcing the generation and download of the adjusted SFC Package based on the SFC Package created by the first user. Finally, the customized SFC Package can be used to be deployed as a new network service.

4.4 Discussion

These three case studies investigate and evaluate different set-ups, , while applying *GENEVIZ*. Both, the construction of a new network service based on selected VNF Packages as well as the adjustment of properties of a certain VNF being part of the SFC, simplify the process for the user compared to existing support solutions. The graphical user interface for the service construction and the chaining recommendation address the critical issue of VNF chaining, helping the user to draft suitable SFCs. Furthermore, by storing the hash of the content of the newly created package on the blockchain, the verification of the package's originality can be performed by any user through *GENEVIZ* as well. All tasks can be performed separately, consuming an unnecessary manual effort for users being inexperienced in the NFV market. The unified thus simplified approach at one single place is achieved by the *GENEVIZ*.

Since this current evaluation is based on case studies only, quantitative evidence on the performance of *GENEVIZ* in terms of intuitiveness and simplification to create SFCs will be performed in future steps. In this sense, a usability evaluation with real users is planned to validate the benefits of *GENEVIZ* and provide quantitative details about the effectiveness, while these evaluations will be based on System Usability Scale (SUS) questionnaires. Also, by constructing SFCs through *GENEVIZ*, specific quantitative gains can be articulated through measurements in a real-world deployment.

The application of a blockchain ensures data integrity of a previously created SFC Package. The block time is assumed to be within in a reasonable amount of time for the end-user, which is not always guaranteed for certain blockchains (*e.g.*, Ethereum may show block time peaks with up to 30 seconds). Although the integrity of content can be guaranteed through the hash verification, the package can already contain malicious code at the moment of the creation of the new package or during the creation of the VNF Package, being part of the SFC Package downloaded. Hence, *GENEVIZ* provides evidence for the end-users trust in a download package from the Internet, since the author and the content can be verified by using the information available in the blockchain.

5 Summary and Future Work

As NFV becomes technically more mature and its infrastructure widely adopted, the demand for specific network services based on the chaining of different virtualized network functions will increase in the years to come. Thus, this paper introduced *GENEVIZ*, a tool for the generation, validation, and visualization of SFC Packages. The graphical user interface proposed leads to a more intuitive and easier construction of new network services. *GENEVIZ* potentially can also lead to fewer mistakes during the creation of new or the adjustment of existing services, as there are fewer steps required to be taken to generate an SFC Package. Thus, visualizations had been proposed within a single Web application to deal with *(i)* the construction of an SFC by chaining different VNFs through a single, directed, and acyclic graph, *(ii)* the adjustment of VNFD properties of VNFs being part of the SFC, *(iii)* supporting the user to create better SFCs by providing a chaining recommender, and *(iv)* the ability to validate, supported by blockchains, a previously created SFC to check its integrity and origin.

GENEVIZ runs as a Web-based application, which can be deployed on a local machine, hence allowing end-users to create and adjust SFC Packages locally. Although *GENEVIZ* provides the possibility to validate data integrity of the content of a package by using blockchains trust in the individual VNF Packages offered by third-parties is still necessary. All visualizations provided by the solution proposed can foster both the growth of the NFV market and business models introduced by marketplaces for VNF-as-a-Service (VNFaaS). Thus, *GENEVIZ* shows the potential to support not only experienced network operators, but also end-users acquiring VNFs from marketplaces. Therefore, a possible positive effect on a broader adoption of NFV technology may become possible.

As future work, existing visualizations can be expanded in order to support different recommendations for the chaining of VNFs inside the service function chaining by using, for example, the affinity between pairs of VNFs. Also, the VNFD Editor can be made configurable to support the editing of additional properties from the VNF Descriptor. Since input fields in the popup window map directly match the properties of a VNFD, *GENEVIZ* can maintain these inputs, too. SFCs created can be offered to other users through a blockchain-based marketplace for SFCs. Finally, the validation of SFCs can be improved not only to ensure data integrity, but also to increase the level of trust for an SFC Package from an unknown source by scanning the content for malicious content. This can be performed during the construction of an SFC, while VNF Packages are imported, or during the import of an existing SFC Package.

References

1. T. Aste, P. Tasca, T. Di Matteo: Blockchain Technologies: The Foreseeable Impact on Society and Industry. In: IEEE Computer. Vol. 50, September 2017, pp. 18–28.
2. L. Bondan, M. F. Franco, L. Marcuzzo, G. Venancio, R. L. Santos, R. J. Pfitscher, E. J. Scheid, B. Stiller, F. De Turck, E. P. Duarte, A. E. Schaeffer-Filho, C. R. P. d. Santos, L. Z. Granville: FENDE: Marketplace-Based Distribution, Execution, and Life Cycle Management of VNFs. In: IEEE Communications Magazine. Vol. 57, January 2019, pp. 13–19.
3. C. Ware: Information Visualization: Perception for Design; 3rd Edition. Elsevier, June 2012, pp. 1–536.
4. R. A. Eichelberger, T. Ferreto, S. Tandel, P. A. Duarte: SFC Path Tracer: A Troubleshooting Tool for Service Function Chaining. In: IFIP/IEEE Symposium

on Integrated Network and Service Management (IM 2017). Lisbon, Portugal, May 2017, pp. 568–571.

5. ETSI GS NFV-MAN: Network Functions Virtualisation (NFV); Management and Orchestration, December 2014
6. M. F. Franco, R. L. d. Santos, A. Schaeffer-Filho, L. Z. Granville: VISION – Interactive and Selective Visualization for Management of NFV-Enabled Networks. In: IEEE 30th International Conference on Advanced Information Networking and Applications (AINA 2016). Crans-Montana, Switzerland, March 2016, pp. 274–281.
7. V. T. Guimares, C. M. D. S. Freitas, R. Sadre, L. M. R. Tarouco, L. Z. Granville: A Survey on Information Visualization for Network and Service Management. In: IEEE Communications Surveys Tutorials. Vol. 18, July 2015, pp. 285–323.
8. B. Han, V. Gopalakrishnan, L. Ji, S. Lee: Network Function Virtualization: Challenges and Opportunities for Innovations. In: IEEE Communications Magazine. Vol. 53, February 2015, pp. 90–97.
9. M. Bucher, M. Franco, E. Scheid: GENEVIZ Prototype - Source Code. https://gitlab.ifi.uzh.ch/franco/geneviz, last visit May 2019.
10. M. Chiosi and D. Clarke and P. Willis and A. Reid and J. Fegers and M. Bugenhagen and W. Khan and M. Fargano and C. Cui and H. Deng: Network functions virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action. In: SDN and OpenFlow World Congress. Vol. 48. Düsseldorf, Germany, October 2012, pp. 1–16.
11. R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, R. Boutaba: Network Function Virtualization: State-of-the-Art and Research Challenges. In: IEEE Communications Surveys Tutorials. Vol. 18, September 2016, pp. 236–262.
12. S. Nakamoto: Bitcoin: A Peer-to-Peer Electronic Cash System, 2009, https://bitcoin.org/bitcoin.pdf last visit June 2019.
13. I. J. Sanz, D. M. F. Mattos, O. C. M. B. Duarte: SFCPerf: An Automatic Performance Evaluation Framework for Service Function Chaining. In: IEEE/IFIP Network Operations and Management Symposium (NOMS 2018). Taipei, Taiwan, April 2018, pp. 1–9.
14. E. Scheid, B. Rodrigues, B. Stiller: Toward a Policy-based Blockchain Agnostic Framework. In: IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2019). Washington, DC, USA, April 2019, pp. 609–613.
15. L. R. Soles, T. Reichherzer, D. H. Snider: A Tool Set for Managing Virtual Network Configurations. In: IEEE SoutheastCon (SoutheastCon 2016). Norfolk, UK, March 2016, pp. 1–4.
16. Truffle Blockchain Group: Ganache Website. https://truffleframework.com/ganache, last visit May 2019.
17. G. Wood: Ethereum: A Secure Decentralised Generalised Transaction Ledger. In: Ethereum Project Yellow Paper. Vol. 151, January 2014, pp. 1–32.