Efficient Zero-Trust-enabled Service Function Chain Deployment in Multi-Vendor Networks

Danyang Zheng¹, Huanlai Xing¹, Xiaojun Cao¹, and Jie Xu¹

¹Affiliation not available

June 07, 2024
Efficient Zero-Trust-enabled Service Function Chain Deployment in Multi-Vendor Networks

Danyang Zheng, Member, IEEE, Huanlai Xing, Member, IEEE, Xiaojun Cao, Senior Member, IEEE, and Jie Xu, Member, IEEE

Abstract—With the advent of zero trust (ZT) security architectures, vendors can bolster their services’ security by continuously verifying every end-to-end traffic flow through the policy enforcement point (PEP). However, the demand for dedicated hardware substantially hinders the extensive deployment of PEP across networks. Consequently, the PEP-based verification can incur overwhelming cost when accommodating network requests involving a sequence of end-to-end traffic flows such as service function chains (SFCs). In this work, we introduce the concept of ZT as a function (ZTaaF) to minimize the verification cost in PEP-based SFC deployment. The ZTaaF enhances the verification flexibility and saves the verification cost by virtualizing the hardware-based PEP into a software module and enabling the inter-trust among the servers from the same vendor. Based on the ZTaaF paradigm, we define a novel problem called SFC deployment with ZTaaF (SFCZT). After analyzing the challenges in SFCZT, we propose an efficient algorithm based on the layered graph technique named ZTaaF-aware SFC embedding (ZAN). Through thorough mathematical analysis, we demonstrate ZAN’s optimality when the network provides sufficient resources for the incoming SFC request. Extensive simulation results validate ZAN’s optimality under the above assumption and show that ZAN significantly outperforms the benchmarks.

Index Terms—Network function virtualization, service function chain, zero trust security architecture, exact algorithm.

I. INTRODUCTION

The emerging virtualization techniques, such as network function virtualization (NFV) and application virtualization, are proposed to enhance service flexibility, reduce capital expenditures, and save operational expenses [1]. These techniques implement network/computation functions via software-based modules, which are termed service functions (SFs) [2]. Typically, a network service request (NSR) comprises a set of SFs and their corresponding resource demands (e.g., computation, bandwidth, RAM) [3]. To accommodate an NSR, the service provider can concatenate the required SFs into a linear structure called service function chain (SFC) and reserve resources over a shared physical network (PN), whereas multiple vendors might co-exist. In the multi-vendor PN, the physical node that hosts an service function instance (SFI) is termed the SF host point (SoT). The physical path that accommodates an SFC is denoted by the SF path (SFP) [4]. The SFC accommodation process is known as service function chain deployment (SFCD) [5].

The COVID-19 pandemic has recently accelerated the widespread adoption of remote work [6]. Research indicates that this trend likely persists even after the pandemic has been brought under control [7]–[9]. In light of this, ensuring the security of tremendous remote network services against cyber attacks, such as malicious macros, compromised domain controllers, and data loss, has become increasingly important [10]. In response, zero trust (ZT) security architecture has been employed, which emphasizes a “never trust, always verify” approach [11]–[16]. Fig. 1 shows a policy enforcement point (PEP)-based ZT verification architecture. PEPs (a.k.a PEP servers) are verification infrastructures composed of the policy administrator (PA) and policy engine (PE). In practical, the PEP can be the SEPIO platform [17], the QAX zero-trust network access server [18], or the service-based (a.k.a. cloud-based) zero-trust network access [19]. Within a PEP, the verification operations such as threat intelligence, access policy, activity logs are performed by the PE to ensure the traffic flow’s security. Under this paradigm, the end-to-end traffic flow must be re-routed to a PEP, which may incur some detouring costs. The detouring cost incurred from the PEP-based verification can be overwhelming if the request involves a sequence of end-to-end traffic flows such as SFCs.

Specifically, Fig. 2 shows the impact from enforcing the PEP-based verification to the SFCD process. Fig. 2a shows a PN with enough resources, whereas the number beside a link is its bandwidth cost. We assume that an SFC request \( S \Rightarrow f_1 \Rightarrow f_2 \Rightarrow D \) comes with a bandwidth demand of 1, where...
S and D represent the source and destination, respectively. Here, the computation cost of instantiating an SF is 3. When employing the state-of-the-art optimization technique such as the one in [20], the optimal SFP is $S \xrightarrow{f_1} A(f_1) \xrightarrow{f_2} B(f_2) \xrightarrow{f_3} D$ with a deployment cost (i.e., the sum of bandwidth and computation cost) of 12 as shown in Fig. 2b. When PEP-based verification is enforced in Fig. 2c, the SFP is created as $S \xrightarrow{P E P} A(f_1) \xrightarrow{P E P} B(f_2) \xrightarrow{P E P} D$ with an overall cost (i.e., the sum of deployment and verification cost) of 20. As one can see, enforcing PEP-based verification requires every end-to-end traffic flow (i.e., the traffic flow between every pair of SoTs hosting adjacent SFs) re-routing to the PEP and incurs the verification cost. Here, the additional cost $20 - 12 = 8$ is an example of the aforementioned detouring cost. Since SFIs are likely host by geographically-distributed SoTs, the PEP-based verification may lead to an excessive amount of detouring costs. Therefore, a question arises: whether the hardware-based PEP verification should be virtualized; thereby, minimizing the verification cost?

To tackle this question, we introduce the concept of zero-trust as a function (ZTaaF) with the support of state-of-the-art virtualization techniques [21], [22]. ZTaaF virtualizes the group of verification functionalities (e.g., access policy, activity logs) into one software-based module called virtual PEP (vPEP). Similar to other virtual functions, vPEP can be flexibly hosted by SoTs. A vPEP can either be customized for a specific service request or be encapsulated for public usage. Here, we assume that every vPEP is well designed and implemented, the software security will be studied in future works. Accordingly, the SoTs hosting the vPEP can be (i) short-term carrier for a single traffic flow or (ii) long-term carrier for public usage. The long-term carrier is specified as the trusted third party (TTP) [23]. Here, in Fig. 2d, when the computation cost of a vPEP is set to 4 and the nodes C and D are converted to TTPs, the SFP is created as $S \xrightarrow{T T P_C} A(f_1, v P E P) \xrightarrow{T T P_D} D$. The overall cost becomes 16, and the verification cost is $16 - 12 = 4$.

For the first time, this work comprehensively studies the problem of how to effectively reduce resource costs when the ZTaaF paradigm is enforced in the SFC deployment process. This work aims to develop a novel cost optimization model and propose efficient algorithms for tackling the above problem. The examination of virtualization’s impact on flow security is deferred for future investigation. First, we formally introduce the concept of ZT as a function (ZTaaF) and analyze the unique challenges brought by this paradigm in multi-vendor networks. Then, we mathematically formulate the problem of SFC deployment with ZTaaF (SFCZT). Next, we propose a novel augmented graph called ZTaaF-aware SoT-oriented layered graph (ZT-SLG) and an efficient algorithm, namely, ZTaaF-Aware SFC embedding (ZAN) to deal with the optimization problem in SFCZT. From our thorough mathematical analysis, we show ZAN’s optimality when sufficient networking resources are provided for the incoming requests. Extensive simulations are conducted to investigate TTP distributions’ impact and validate ZAN’s optimality, while showing that ZAN significantly outperforms the benchmarks.

The rest of the paper is organized as follows. Section II introduces the concept of ZTaaF and Section III summarizes related work. Section IV defines the SFCZT problem. In Sections V and VI, we analyze SFCZT and propose our techniques and algorithms. Section VII shows the experimental results. We conclude our work in Section VIII.

### II. Zero Trust as a Function (ZTaaF)

The ZTaaF paradigm leverages the virtualization techniques to implement the hardware-based PEP into a software-based module called virtual PEP (vPEP) [21]. This enables the flexible accommodation of PEP-based verifications. Without loss of generality, we assume that if a node (or server) is selected to serve as a TTP, it can host no SF instances but the instance of vPEP. Meanwhile, considering the fact that multiple vendors may co-exist in a PN to provide virtualization-enabled services, the ZTaaF paradigm also enables the inter-trust among servers belonging to the same vendor. In other words, no verification is further needed if two servers are from the same vendor under the paradigm of ZTaaF. With vPEP and inter-trust among homo-vendor servers, the ZTaaF paradigm can reduce the overhead associated with security verifications and increases the cost-efficiency when deploying requests associated with a sequence of end-to-end traffic flows.

Table I lists the important notations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSR</td>
<td>Network service request.</td>
</tr>
<tr>
<td>SFC</td>
<td>Service function chain.</td>
</tr>
<tr>
<td>SFP</td>
<td>Service function path.</td>
</tr>
<tr>
<td>SFI</td>
<td>Service function instance.</td>
</tr>
<tr>
<td>SoT</td>
<td>SF host point.</td>
</tr>
<tr>
<td>ZTaaF</td>
<td>Zero trust as a function.</td>
</tr>
<tr>
<td>SFCZT</td>
<td>SFC deployment with zero trust as a function.</td>
</tr>
<tr>
<td>TTP</td>
<td>Trusted third party.</td>
</tr>
<tr>
<td>PEP</td>
<td>Policy enforcement point.</td>
</tr>
</tbody>
</table>
In ZTaaF paradigms, there are three verification options: (i) homo-vendor SoTs (homo-vendor-based verification); (ii) nearby TTP detour (TTP-based verification); and (iii) vPEP instantiation (vPEP-based verification). Fig. 3 addresses the differences among these three when handling an SFC request with two SFs: “feature extraction” and “model training”. Fig. 3a displays an example of deploying the SFC using the homo-vendor-based verification. With inter-trust from homo-vendor servers, no verification is needed between these two SoTs when they host adjacent SFs and are from the same vendor. In Fig. 3a, since SoT A and B are homo-vendor, the data exiting the SFI of “feature extraction” can be directly transmitted to SoT B and processed by “model training”. As one can see, since no additional verifications are needed, homo-vendor-based verification incurs no verification cost.

Fig. 3b demonstrates an instance of deploying the SFC via the TTP-based verification. This method utilizes the vPEP that has been deployed over the TTP for verification and is defined as below: For two SoTs hosting a pair of adjacent SFs, the SoT hosting the former SF must detour the traffic flow through a nearby TTP before reaching the other SoT. In Fig. 3b, the outsourced traffic flow from SoT A has to be routed to a TTP for verification before reaching SoT B, which may incur additional routing costs for verification purposes.

Fig. 3c employs the vPEP-based verification. This method leverages the flexibility of vPEP’s implementation and is defined as: For two SoTs hosting a pair of adjacent SFs, the former SoT must instantiate a vPEP instance and verify the traffic flow via this vPEP before sending it out. In Fig. 3c, a vPEP instance is instantiated following the “feature extraction" at SoT A. Here, the verification process is done by installing a vPEP over the SoT that host client’s SFs, and the verification cost comes from the additional computation at SoT A.

As one can see that, to determine which ZTaaF verification methods are available for two SoTs, one must check whether these two SoTs are from the same vendor or not. We use Fig. 4 to further depict the concepts of homo-vendor SoTs and assume that every SoT is composed of a computation component and a routing component. In Fig. 4, the SoTs A and B host the adjacent SFs and both are from Vendor_A, while the forwarding path between them goes through an SoT C that is from Vendor_B. In this case, the SoT C acts like an intermediate router, whereas only the routing component of SoT C will be utilized and no change will be made on the traffic flow’s payloads. Since no data processing operations will be performed at SoT C, no ZTaaF verification is needed. Thus, SoTs A and B are regarded as homo-vendor.

Even though the ZTaaF-based verifications bring flexibility in accommodating SFCs, they are not without their challenges. Here, we analyze the potential challenges in each ZTaaF-based verification methods: (i) The homo-vendor-based verification incurs no verification cost only when all SFs are hosted by the servers from the same vendor. However, if an SFC requests ChatGPT and DNS, the “ChatGPT” service likely can only be legally provided via servers from OpenAI while the “DNS” may be hosted by vendors other than OpenAI (e.g., Cisco or Google). Hence, to accommodate multiple SFs requested by an SFC, very likely, multiple vendors will be involved. Moreover, homo-vendor SoTs are typically located in geographically separate areas (e.g., for the purpose of load balancing or failure protection), to map all SFs from an SFC onto these home-vendor SoTs may introduce significant routing costs for connections. (ii) The additional routing cost for the TTP-based verification depends on the distances among SoTs and the nearby TTP. When only a limited number of servers are selected to serve as TTPs (e.g., due to the budget constraint), the verification cost incurred by detouring the traffic flow to TTPs might be substantial. (iii) The vPEP-based verification can be flexible, but installing a vPEP over an SoT will consume computation resources at the corresponding SoT, lowering the computation availability of the PN.

As one can see that, every ZTaaF-based verification method has its pros and cons. In this work, we comprehensively investigate how to cost-effectively accommodate an SFC while taking advantages of the above verification methods. As to be shown in the following sections, we propose efficient mathematical model, algorithms, and analysis for optimizing the SFC deployment process in multi-vendor networks.
III. RELATED WORKS

In this section, we summarize the works that are related to SFC deployment (SFCD) with security requirements. As shown in Fig. 5, there are two broad categories: (i) security-aware SFC deployment, and (ii) trust-aware SFC deployment. For the former type, a numerical security level constraint is required by the client, and the service provider must create an SFC with the security level higher than the constraint. For the latter type, the security is ensured by employing the trust verification methods such as ZT or ZTaaF.

Security-aware SFC deployment: The authors in [24] did a thorough survey on the importance of enhancing security in delivering SFCs. The authors in [25]–[29] proposed various methodologies on deploying SFCs while meeting the specified security requirements. In specific, the authors in [25], the authors assumed that every SF is associated with a pre-given security level and investigated a cost optimization problem of meeting a given security guarantee by selecting and embedding a proper set of SFs. Similarly, the authors in [26] assumed that the security level of each SF is pre-defined and investigated the resource optimization problem of accommodating multiple security-aware SFC requests in a resource-sharing manner. In [27], taking service level agreement (SLA) into account, the authors introduced an on-site function named evidence bloom filter (eBF) that records adversarial behavior of vendor devices, and the service provider can employ the proposed query technique on eBFs to measure the quality of service (QoS) for an SFC. When considering network security defense patterns, the authors in [29] proposed a multi-stage security defense pattern aware placement (SDPAP) algorithm to minimize the SFC deployment cost. Most existing works on security-aware SFC tried to deploy SFC requests while meeting a pre-defined security requirements, whereas the security verification requirement is not enforced.

ZT-based SFC deployment: In the literature, few works have focused on employing ZT architectures to facilitate the deployment of network services [30]–[34]. In [30], the authors employed the ZT architecture to help fulfill the SFC security requirements while maintaining SFC’s flexibility. In [31], the authors utilized the virtualization technique to implement the PEP with a software-based module called system flow (Sysflow) controller that is able to host various security-related SFs. The authors in [32] employed the ZT network architecture to enhance the security of any network services accessing the virtual power plants and demonstrated its advantages compared to the traditional Purdue model. The authors in [33] proposed a survivable ZT architecture with distributed control components, which can guarantee network services’ verification while taking the cyber resilience into account. In [34], the authors proposed a network architecture called zero trust service function chaining (ZTSFC) to improve the performance. All above works enforce the ZT-based verification by requiring the network requests to be verified via the dedicated hardware, whereas the dependency of such hardware may lead to network ossification.

ZTaaF-based SFC deployment: The most related work is [21]. In [21], the authors proposed a virtual evolved packet core-virtual software defined perimeter (vEPC-vSDP) framework to provide secure communications, whereas vEPC will perform the function of PEP to provide zero-trust environment. Compared to [21], for the first time, we investigate how to employ the virtualized PEP structure to efficiently satisfy the network requests that incorporate a sequence of end-to-end traffic flows. We propose an efficient algorithm based on the layered graph technique named ZTaaF-aware SFC embedding (ZAN). The proposed ZAN approach does not need to meet a specified security requirement, while effective verification methods are proposed and employed.

IV. SERVICE FUNCTION CHAIN DEPLOYMENT WITH ZERO TRUST AS A FUNCTION (SFCZT)

A. Physical Network (PN) Model

A PN is denoted by an undirected graph \( G = (N, L, F) \), where \( N, L, \) and \( F \) represent the set of physical nodes, physical links, and installable SFIs, respectively. \( N \) consists of two sets \( S \) and \( T \) \((N = \{S, T\})\), which represent the set of SoTs and TTPs, respectively. Each SoT \( s \in S \) can host a set of SFIs \( \mathcal{F}_n \) \((\mathcal{F}_n \subseteq F)\) and the instances of vPEP. Meanwhile, each SoT has a computing capacity \( C_s \) and belongs to one specific vendor \( s_{ven} \). Each TTP \( t \in T \) has installed a set of vPEP instances that can be employed for public usage and cannot host any other SFs. Each physical link \( l \in L \) has a weight \( w_l \) and a bandwidth capacity of \( bw_l \). Let \( P \) denote the set of paths in the PN. We use \( r_{m,n}^i \) to denote the \( i^{th} \) path from \( m \) to \( n \). The normalized computation cost at node \( n \) and bandwidth are denoted by \( C_{cpu,n} \) and \( C_{bw} \), respectively.

B. Network Service Request Model

A network service request (NSR) is denoted by a 4-tuple \( NSR =< V, SFC, SFC_{cpu}, SFC_{bw} > \), where \( V \) includes the set of required SFs, \( SFC \) is the executing order, \( SFC_{cpu} \) includes the computation demands of each SF, and \( SFC_{bw} \) represents the set of bandwidth demands between every pair of adjacent SFs. In specific, an SF can be represented by \( SFC = \{f_0, f_1, ..., f_i, ..., f_{\tau}, f_{\tau+1}\} \), where \( f_i \) represents the \( i^{th} \) SF in the SFC and \( i \in [1, \tau] \); \( f_0 \) and \( f_{\tau+1} \) represent source (S) and destination (D), respectively. Correspondingly, \( SFC_{cpu} = \{c_1, c_2, ..., c_{\tau}\} \) and \( SFC_{bw} = \{bw_{0,1}, bw_{1,2}, ..., bw_{\tau+1}\} \). We denote a pair of adjacent SFs by \( u, v \in V \), and the computation demands of deploying a vPEP is denoted by \( c_{cpu} \).
C. Problem Formulation

The optimization problem of SFCZT is defined as: given an NSR, how to accommodate this NSR over a PN such that (i) the following constraints are satisfied, and (ii) the overall deployment and verification cost is minimized. To facilitate the presentation, the important abbreviations are listed in Table II.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>Set of nodes including $S$ and $T$.</td>
</tr>
<tr>
<td>$S$</td>
<td>Set of SF host points (SoT).</td>
</tr>
<tr>
<td>$T$</td>
<td>Set of trust third party (TTP).</td>
</tr>
<tr>
<td>$u, v$</td>
<td>Two physical nodes from $N = {S, T}$.</td>
</tr>
<tr>
<td>$m, n$</td>
<td>Two SoT $m$ and $n$.</td>
</tr>
<tr>
<td>$p_{m,n}$</td>
<td>The $l^{th}$ path connecting $m$ and $n$.</td>
</tr>
<tr>
<td>$\tau$</td>
<td>The number of required SFs.</td>
</tr>
<tr>
<td>$l$</td>
<td>A physical link $l$.</td>
</tr>
<tr>
<td>$M_n$</td>
<td>$= 1$, $v$ is mapped over $n$; $= 0$, otherwise.</td>
</tr>
<tr>
<td>$M_{m,n}$</td>
<td>$= 1$, a flow for $u, v$ is hosted by $p_{m,n}$; $= 0$, otherwise.</td>
</tr>
<tr>
<td>$M_l^{u,v}$</td>
<td>$= 1$, a flow for $u, v$ is hosted by $l$; $= 0$, otherwise.</td>
</tr>
<tr>
<td>$H_{m,n}$</td>
<td>$= 1$, if $m$ and $n$ are from different vendors; $= 0$, otherwise.</td>
</tr>
<tr>
<td>$\delta_{m,n}^{u,v}$</td>
<td>A verification flow for $u, v$ over $n$ passes $l$; $= 0$, otherwise.</td>
</tr>
<tr>
<td>$\delta_{m,n}^{p_{m,n}}$</td>
<td>A verification flow for $u, v$ uses $p_{m,n}$; $= 0$, otherwise.</td>
</tr>
<tr>
<td>$\delta_{m,n}^{l}$</td>
<td>$= 1$, a vPEP is placed after $u$ at $m$; $= 0$, otherwise.</td>
</tr>
</tbody>
</table>

Eqs. (8)-(11) demonstrate decision variables of TTP-based verification ($\Delta$) and vPEP-based verification ($\delta$).

$$\Delta_{u,v}^{m,n} = \begin{cases} 1, & \text{a verification flow for } u, v \text{ over } m, n \text{ bypasses } t; \\ 0, & \text{otherwise.} \end{cases}$$

$$\Delta_{u,v}^{p_{a,b}} = \begin{cases} 1, & \text{a verification flow for } u, v \text{ uses } p_{-a,b}, a, b \in N; \\ 0, & \text{otherwise.} \end{cases}$$

$$\delta_{m}^{u,v} = \begin{cases} 1, & \text{a vPEP is placed after } u \text{ at } m; \\ 0, & \text{otherwise.} \end{cases}$$

Eq. (12) constrains that if SoTs $m$ and $n$ are not homo-vendor, then either the TTP-based verification or the vPEP-based verification has to be performed between them.

$$\mathbb{H}_{m,n} \cdot (M_{m}^{u,v} + M_{n}^{u,v} - 1) \leq \Delta_{m,n}^{u,v} + \delta_{m}^{u,v}, \forall u, v, \forall m, n \in S \quad (12)$$

Eq. (13) ensures that if $\Delta_{m,n}^{u,v} = 1$, then there must exist two paths connecting $m$ and $n$ to $t$, respectively.

$$\Delta_{p_{m,n}}^{u,v} \leq \Delta_{m,n}^{u,v} + \Delta_{l}^{u,v} - 1, \forall u, v, \forall t \in T, \forall m, n \in S \quad (13)$$

Eq. (14) ensures that if a path $i$ is employed for TTP verification, then the link $l$ along this path must also be employed.

$$\Delta_{l}^{u,v} \leq \Delta_{p_{m,n}}^{u,v}, \forall u, v \in V, \forall a, b \in N, \forall p_{a,b} \in P \quad (14)$$

Resource Reservation Constraints: Eqs. (15)-(16) show the computation and bandwidth capacity constraint on each node and link, respectively.

$$\Sigma_{v \in V} (M_{m}^{v} \cdot c_{v} + \delta_{v}^{u,v} \cdot c_{uv}) \leq C_{v}, \forall v \in S \quad (15)$$

$$\Sigma_{u,v \in V} (M_{l}^{u,v} + \Delta_{l}^{u,v}) \cdot bw_{uv,v} \leq bw_{v}, \forall l \in L \quad (16)$$

Eqs. (17)-(18) calculate the overall normalized computation cost and bandwidth cost, respectively.

$$\Sigma_{v \in S} M_{m}^{v} = 1, \forall v \in V \quad (5)$$

$$\Sigma_{p_{m,n} \in P} M_{m,n}^{u,v} \geq M_{m}^{u,v} + M_{n}^{u,v} - 1, \forall u, v \in V, \forall m, n \in S \quad (6)$$

$$\Sigma_{M_{m,n}^{u,v}, v \in V, \forall l \in p_{m,n}, \forall p_{m,n} \in P \quad (7)$$

ZTaF-based verification Constraints: To begin, we employ a given parameter $H_{m,n}$ to address whether SoTs $m$ and $n$ are homo-vendor or not ($= 0$, $m_{ven} = n_{ven} = 1$, otherwise).

V. Layered Graph for SFCZT

Although the cost optimization problem of SFCZD is NP-hard, existing works have shown that the layered graph can obtain the SFP with the least deployment cost when network resources are sufficient for the incoming SFC request. In the following subsections, we analyze two different layered graph techniques: PN-oriented layered graph [36], and SoT-oriented layered graph [20], for the cost optimization in SFCZT.
A. PN-oriented Layered Graph for SFCZT

Given a PN and an NSR, a prevalent approach to obtain the least-deployment-cost SFP is to employ the PN-oriented layered graph-based technique [36]. The PN-oriented layered graph is created by: (I) generating \( \tau + 1 \) replicas of PN, and (II) adding an vertical edge\(^1\) between every two adjacent PN replicas if the SoTs in the \( i^{th} \) replica can host the SFI of \( f_i \). Here, every vertical edge is associated with a weight representing the computation cost for the corresponding SF. Once the PN-oriented layered graph is generated, the least-deployment-cost SFP is the shortest path connecting the source in the first layer to the destination in the last layer [36].

![PN-oriented layered graph](image)

Fig. 6: A multi-vendor PN with an NSR of \( A \Rightarrow f_1 \Rightarrow f_2 \Rightarrow E \). The computation cost of a vP EP is 4.

Next, we employ the PN and NSR in Fig. 6 to show the generation process of a PN-oriented layered graph. In Fig. 6, the SoTs in the same color are from the same vendor (i.e., A, C and D belong to Vendor_A, while B and E are from Vendor_B), and the number beside each link shows its cost. When inputting the PN and the NSR in Fig. 6 to the aforementioned process, Fig. 7a shows the generated PN-oriented layered graph. Note that, the number (in the dashed square) beside each vertical edge is the computation cost of the corresponding SF. After obtaining the PN-oriented layered graph as Fig. 7a, the least-cost SFP is the shortest path connecting \( A_1 \) to \( E_3 \): \( A_1 \rightarrow B_1(f_1) \rightarrow B_2 \rightarrow C_2(f_2) \rightarrow C_3 \rightarrow E_3 \) as demonstrated by the red arrows in Fig. 7b.

![Least-cost SFP](image)

Fig. 7: An example of generating the least-deployment-cost SFP from the PN-oriented layered graph.

Accordingly, this process did not consider the ZTaaF-based verifications and the identified SFP may not meet the ZTaaF-based verification constraints as shown in Eqs. (8)-(14). For example, the connections between \((A_1, B_1), (B_2, C_2)\), and \((C_3, E_3)\) in Fig. 7b violate the ZTaaF-based verification constraints. Intuitively, to lower the overall cost while meeting the ZTaaF-based verification constraints, the following operations can be taken to create the SFP: (i) creating an incremental solution based on the least-deployment-cost SFP obtained via the traditional PN-oriented layered graph such as Fig. 7b, or (ii) taking into account of the ZTaaF-based verification cost during the construction process so that the shortest path meets the ZTaaF-based verification constraints.

Operation (i) can be achieved when we employ the cheapest ZTaaF-based verification method between the endpoints of each violated connection. Due to the local optimality brought by this incremental solution, the created SFP likely loses the optimality regarding the overall cost. For example, when assuming the computation cost of a vP EP is 4, by employing the above incremental solution, one can employ the path of \( A_1 \rightarrow TTP_1 \rightarrow B_1(f_1) \rightarrow B_2(vP EP) \rightarrow C_2(f_2) \rightarrow C_3(vP EP) \rightarrow E_3 \) as shown in Fig. 8a. Compared to the SFP in Fig. 7b, a TTP is employed between \( A_1 \) and \( B_1 \), while the other two violated connections are fixed via installing a vP EP. Consequently, the overall cost of the SFP becomes 17. However, one can employ the other SFP \( A_1 \rightarrow B_1 \rightarrow D_1(f_1) \rightarrow D_2 \rightarrow C_3(f_2) \rightarrow C_3(vP EP) \rightarrow E_3 \) with the overall cost of 14 as shown in Fig. 8b. Here, \((A_1, D_1)\) and \((D_2, C_2)\) employ the homo-vendor verification, while \((C_3, E_3)\) utilizes the vP EP-based verification. Hence, we find that the local optimality from the incremental solutions cannot guarantee a global optimality of the overall cost in SFCZT.

Operation (ii) requires properly associating the ZTaaF-based verification cost to either edges or nodes. In each layer, the traffic flow starts from an ingress SoT and ends at an egress SoT. Here, the ingress and egress SoTs represent the node that the traffic flow starts and ends at each layer. Specifically, in Fig. 8b, \( A_1 \) and \( D_1 \) are the ingress and egress SoTs of the 1\(^{st}\) layer. Whether these two SoTs are homo-vendor or not dictates which ZTaaF-based verification method will be employed; thereby affecting the ZTaaF-based verification cost. However, before the shortest path is identified, it remains unclear which nodes will be the ingress and egress SoTs at each layer. Hence, the above dilemma makes operation (ii) barely feasible.

![Analysis of operation (ii)](image)

Fig. 8: Analysis of employing operation (i) to the PN and the NSR in Fig. 7. The number in the parenthesis represents computation cost, while the others denote routing cost.
B. SoT-oriented Layered Graph for SFCZT

The other approach of obtaining the least-deployment-cost SFP is the SoT-oriented layered graph (SLG) [20]. The SoT-oriented layered graph can be generated in three steps: (I) creating an empty layered graph $SLG = (\mathbb{N}, \mathbb{E})$ with $\tau$ + 2 layers, whereas $\mathbb{N} = \{N_0, N_1, ..., N_{\tau+1}\}$ and $\mathbb{E} = \{E_{0,1}, E_{1,2}, ..., E_{\tau,\tau+1}\}$ represent the set of nodes and edges in the layered graph, while $N_0 = S$ and $N_{\tau+1} = D$; (II) for the $i^{th}$ layer ($i \in [0, \tau]$), adding all physical candidates $1$ of $f_i$ to $N_i$; and (III) for the $i^{th}$ layer ($i \in [0, \tau]$), connecting each physical candidate to all physical candidates in the $(i+1)^{th}$ layer via a directed edge. Fig. 9 shows an example of an SLG for the SFC $S \rightarrow f_1 \rightarrow f_2 \rightarrow D$. In Fig. 9, $\mathbb{N} = \{N_0, N_1, N_2, N_3\}$, whereas $N_0 = S$, $|N_1| = x$, $|N_2| = y$, and $N_3 = D$. Note that, we employ $[N_i]$ to represent the cardinality of $N_i$. For convenience, in the SoT-oriented layered graph, the $j^{th}$ node at the $i^{th}$ layer is denoted by $n_j^i$. Every edge is associated with a weight $w_{i,\alpha,\beta}^{j+1}$ ($i \in [0, \tau]$, $\alpha \in [1, |N_i|]$, $\beta \in [1, |N_{\alpha+1}|]$), which represents the cost between $n_\alpha^i$ and $n_\beta^{i+1}$. This cost can either be the deployment cost, verification cost or overall cost.

![Diagram](image)

**Fig. 9:** An SLG for an SFC $S \Rightarrow f_1 \Rightarrow f_2 \Rightarrow D$.

Different from the PN-oriented layered graph, in an SLG, an edge's weight can represent any cost between two SoTs. When the edge's weight is referred to as the overall cost of the shortest path between $m$ and $n$.

$$w_{i+1,\alpha,\beta}^{i+1} = \min (c_{f_i} + C_{cpu}^\alpha) + (\|p_{n_\alpha^i, n_\beta^{i+1}}\| \times \text{bw}_{i+1,\alpha,\beta}^\alpha \times C_{bw}) \quad (19)$$

When sufficient resources are provided for the incoming SFC request, the least-deployment-cost SFP is the shortest path connecting source and destination in the SLG [20]. Note that, when the edge's weight is referred to as the overall cost (including deployment and verification cost), three types of edges in SLG can be defined correspondingly to different ZTaaS-based verification methods: (i) homo-vendor edge, (ii) TTP-based edge, and (iii) vPEP-based edge. An edge's weight represents the overall cost of employing the corresponding ZTaaS-based verification method.

**Homo-Vendor Edge:** The homo-vendor edge connects two adjacent homo-vendor SoTs in the SLG. Since no specific verification process is needed, the weight of a homo-vendor edge can directly reflect the deployment cost, which is the same as Eq. (19). Accordingly, one can create the Homo-Vendor-edge-based SoT-oriented Layered Graph (HV-SLG) by replacing the above step (III) to: for the $i^{th}$ layer ($i \in [0, \tau]$), connecting each physical candidate to all homo-vendor candidates in the $(i+1)^{th}$ layer via a directed edge, the weight of which is calculated via Eq. (19). Fig. 10a shows an example of HV-SLG, whereas the number beside each edge is calculated via Eq. (19). Notably, homo-vendor edges can only be created between two homo-vendor SoTs such as $(A_0, D_1)$, $(D_1, C_2)$.

**TTP-based Edge:** The TTP-based edge enforces every pair of adjacent SoTs from different vendors to re-route and connect via a nearby TTP. The weight of a TTP-based edge includes the deployment and TTP-based verification cost. Eq. (20) calculates the weight of a TTP-based edge, where the least-routing-cost TTP for $n_\alpha^i$ and $n_\beta^{i+1}$ will be selected.

$$w_{i+1,\alpha,\beta}^{i+1} = \min (c_{f_i} + C_{cpu}^\alpha + \|p_{n_\alpha^i, n_\beta^{i+1}}\| \times \text{bw}_{i+1,\alpha,\beta}^\alpha \times C_{bw}) \quad (20)$$

Accordingly, one can create the TTP-edge-based SoT-oriented Layered Graph (TP-SLG) by incorporating Eq. (20) into the aforementioned step (III). Fig. 10b demonstrates an example of TP-SLG, whereas the number beside each edge is calculated via Eq. (20). Note that, even if $A_0$ and $D_1$ are homo-vendor, the TTP-edge will enforce the TTP-based verification, and the shortest path $A \rightarrow TTP \rightarrow B \rightarrow D$ with the routing cost of 4 will be employed between $A_0$ and $D_1$.

**vPEP-based Edge:** The vPEP-based edge requires a pair of SoTs to install a vPEP. Due to this, the weight of a vPEP-based edge needs to incorporate (i) the deployment cost, and (ii) the vPEP installation cost. Eq. (21) demonstrates the weight calculation of a vPEP-based edge, whereas a vPEP is instantiated at the upper-layer SoT $n_\alpha^i$.

$$w_{i+1,\alpha,\beta}^{i+1} = \min (c_{f_i} + C_{cpu}^\alpha + \|p_{n_\alpha^i, n_\beta^{i+1}}\| \times \text{bw}_{i+1,\alpha,\beta}^\alpha \times C_{bw}) \quad (21)$$

Accordingly, one can create the vPEP-based SoT-oriented Layered Graph (vP-SLG) by incorporating Eq. (21) into the aforementioned step (III). Fig. 10c shows an example of vP-SLG, whereas the number beside each edge is calculated via Eq. (21). Notably, every pair of SoTs in vP-SLG is required to install a vPEP even if these two SoTs are homo-vendor. For example, although both $D_1$ and $C_2$ are from Vendor_A, a vPEP is built at the upper-layer SoT $D_1$. By doing so, the computation cost is 3 ($f_3$) + 4 (vPEP) = 7, while the routing cost is 1 ($D_2\rightarrow C_2$), resulting in the overall cost of 8.

Next, assuming that the physical network has sufficient resources to host the incoming SFC, we prove the following theorems and lemmas. In specific, we formally define what is a PN with sufficient resources for an incoming SFC request. We assume that an SFC with two SFs and three virtual links, each of which requires $|cpu|$ CPU cores and $|bw|$ bandwidth, respectively. When every physical candidate of these two functions can provide $2 \times |cpu|$ CPU cores (i.e., simultaneously host these two SFs) and every link can provide $3 \times |bw|$ bandwidth availability (i.e., to host the three virtual links in the SFC), the PN provides sufficient resources for this SFC. Here, as the links without enough bandwidth can be pruned from the PN, the SFC deployment process only takes the
unpruned links into account. To summarize, in a PN with sufficient resources for the incoming SFC request with $|\tau|$ SFs and $|\tau| + 1$ virtual links, every physical candidate $n$ provides CPU cores of $|\mathcal{V}_{\text{PN}}^n| * |\mathcal{P}_n|$, while the links without $|\mathcal{V}_{\text{PN}}^n| * |\mathcal{P}_n|$ bandwidth are pruned. Note that, since the resource demands from a specific SF request normally are far less than the available resources provided by a PN [37], this is a mild assumption. In our proofs, we employ $p_{a,u+1}$ to represent a subpath along the optimal SFP, where physical nodes $a$ and $b$ host SF $u$ and $u + 1$, respectively. Since $p_{a,u+1}$ has been identified, it must have employed one of the ZTaaF-based verification methods. Thus, in the corresponding SLG, let $n_u \rightarrow a$ and $n_{u+1} \rightarrow b$, there must be an edge connecting $n_u^y$ to $n_{u+1}^y$ that corresponds to $p_{a,b}$. For example, let $D \rightarrow C$ be the subpath hosting $f_1 \Rightarrow f_2$ along the optimal SFP. The $n_2^y$ ($D_1$) and $n_2^y$ ($C_2$) in HV-SLG (Fig. 10a) correspond to physical nodes $D$ and $C$, respectively. Accordingly, the edge between $n_2^y$ and $n_2^y$ in HV-SLG can serve as the subpath $p_{C,D}$. Table III lists the necessary notations.

### TABLE III: Notations for Proofs.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLG</td>
<td>SoT-oriented layered graph.</td>
</tr>
<tr>
<td>HV-SLG</td>
<td>Homo-vendor-edge-based SoT-oriented layered graph.</td>
</tr>
<tr>
<td>TTP-SLG</td>
<td>TTP-edge-based SoT-oriented layered graph.</td>
</tr>
<tr>
<td>vP-SLG</td>
<td>vPEP-edge-based SoT-oriented layered graph.</td>
</tr>
<tr>
<td>ZT-SLG</td>
<td>ZTaaF-aware SoT-oriented layered graph.</td>
</tr>
<tr>
<td>$N$</td>
<td>The set of nodes in an SLG.</td>
</tr>
<tr>
<td>$E$</td>
<td>The set of edges in an SLG.</td>
</tr>
<tr>
<td>$p_{a,b}$</td>
<td>The shortest path between physical nodes $a$ and $b$, whereas $a$ and $b$ host SF $u$ and $u + 1$, respectively.</td>
</tr>
<tr>
<td>$n_u^x$, $n_{u+1}^y$</td>
<td>The $x$th and $y$th SFs in the $u$th and $(u + 1)$th layers.</td>
</tr>
<tr>
<td>$C_{a,b}$</td>
<td>The overall cost of $p_{a,b}$.</td>
</tr>
<tr>
<td>$C_{a,b}$</td>
<td>The overall cost of $p_{a,b}$ along the optimal SFP that has less overall cost than the edge connecting $n_u^x$ to $n_{u+1}^y$ in the corresponding SLG, and this theorem stands.</td>
</tr>
</tbody>
</table>

**Theorem 1.** When network resources are sufficient for the incoming SFC, the overall cost of a subpath $p_{a,b}$ along the optimal SFP must not be less than the weight of the edge connecting $n_u^x$ and $n_{u+1}^y$ in the corresponding SoT-oriented layered graph, whereas $n_u^x = a$ and $n_{u+1}^y = b$. **Proof.** We prove this theorem via contradictions. We assume that there exists a path $p_{a,b}$ along the optimal SFP that has less overall cost than the weight of the edge connecting $n_u^x$ and $n_{u+1}^y$ in the corresponding SLG. According to Eqs. (19)-(21), the weight of an edge is decided by (i) the upper-layer SoT $n_u^x$ and the type of the required function, and (ii) the shortest path for connecting $n_u^x$ and $n_{u+1}^y$. Since the SoT and the type of the function has been specified, the overall cost is then totally determined by the shortest path. Under this circumstance, if the overall cost of the subpath $p_{a,b}$ is less than the weight of the edge in the corresponding SLG, then the shortest path algorithm does not generate the path with the least weight, whereas the contradiction arises. Consequently, there does not exist such a subpath $p_{a,b}$ along the optimal SFP that has less overall cost than the edge connecting $n_u^x$ to $n_{u+1}^y$ in the corresponding SLG, and this theorem stands. 

**Lemma 1.** When network resources are sufficient for the incoming SFC, any subpath $p_{a,b}$ along the optimal SFP selects the verification method incurring the least overall cost. **Proof.** We prove this lemma via contradictions. We assume that there exists a subpath $p_{a,b}$ along the optimal SFP but does not select the verification method incurring the least overall cost. For clarity, if this subpath refers to $C \rightarrow D$, all three types of verification methods can be selected. Let vP-SLG be the selected one, then the other two SLGs are the unselected SLGs. To facilitate the proof, we employ $C_{a,b}$, $C_{a,b}$, and $C_{a,b}$ to denote the overall cost of the subpath $p_{a,b}$, the weight of the corresponding edge in the selected SLG, and the weight of the corresponding edge in the unselected SLG, respectively. Considering that the unselected verification method can incur less overall cost ($C_{a,b}$) of connecting $a$ and $b$ than the one that $p_{a,b}$ has selected, Eq. (23) holds based on Theorem 1.

$$C_{a,b} < C_{a,b} \leq C_{a,b}$$  \hspace{1cm} (22)

Hence, substituting this less-cost edge with $p_{a,b}$ can create an SFP with less overall cost than the “optimal” one, which incurs a contradiction. Consequently, any subpath $p_{a,b}$
along the optimal SFP must select the verification method incurring the least overall cost, and this lemma stands.

Based on Theorem 1 and Lemma 1, Theorem 2 holds.

**Theorem 2.** When network resources are sufficient for the incoming SFC, the overall cost of any sub-path \( p_{u,b} \) along the optimal SFP must be equal to the weight of the least-weight edge connecting \( n_u \) and \( n_{b+1} \) among HV-SLG, TP-SLG, and vP-SLG, whereas \( n_u^x = a \) and \( n_{b+1}^x = b \).

Again, an optimal SFP can be regarded as a sequence of consecutive sub-paths \( (p_{u,b}^{i+1}, u \in [0, \tau], a, b \in S) \). The overall cost of every sub-path must correspond to one edge among HV-SLG, TP-SLG, and vP-SLG at the corresponding layer; otherwise, this sub-path does not belong to the optimal SFP based on Theorem 2. Following this, we define a novel SLG structure called ZTaaf-aware SoT-oriented Layered Graph (ZT-SLG), which has the same set of nodes as \( N \) but incorporates all edges from HV-SLG, TP-SLG, and vP-SLG. In specific, \( ZT-SLG = (N, E_{ZT}) \), \( E_{ZT} = E_{HV} \cup E_{TP} \cup E_{vP} \), whereas \( E_{HV} \), \( E_{TP} \), and \( E_{vP} \) represent the set of edges in HV-SLG, TP-SLG, and vP-SLG, respectively. Fig. 10d shows an example of ZT-SLG, which merges all edges from Figs. 10a-10c. With the analysis above, the following theorem holds.

**Theorem 3.** When network resources are sufficient for the incoming SFC, the shortest path connecting source to destination in ZT-SLG is the least-overall-cost SFP.

VI. ZTaaf-Aware SFC Embedding (ZAN)

Based on techniques and analysis above, we now propose the ZTaaf-aware SFC embedding (ZAN) algorithm including (i) ZT-SLG generation, and (ii) shortest path identification.

**Algorithm 1 ZT-SLG Generation (ZSG)**

1. **Inputs:** \( G, NSR \);
2. **Output:** \( ZT-SLG = (N_{ZT}, E_{ZT}) \);
3. Initialize \( N_{ZT} \) \((x \in [0, \tau + 1])\) and \( E_{ZT} \) \((y \in [0, \tau])\) as \( \emptyset \);
4. Set \( i \) to 0;
5. **while** \( i \leq \tau + 1 \) **do**
6. \( N_i^{ZT} = \{n|f_i \in F_n\} \)
7. **if** \( i \neq 0 \) **then**
8. **for** Every pair of SoTs \( n_i \) and \( n_i \) **do**
9. Get the weight of edge \( w_{i,i} \) via Eq. (23);
10. \( E_{ZT} = E_{ZT} + w_{i,i} \);
11. **end for**
12. **end if**
13. \( N_{ZT} = N_{ZT} \cup N_i^{ZT} \);
14. **end while**
15. **Return:** \( ZT-SLG = (N_{ZT}, E_{ZT}) \);

From our observation, ZT-SLG is a directed acyclic graph (DAG) [38]. Between every two adjacent layers, only one edge that connects two SoTs will be employed to carry the traffic flow; otherwise, ZT-SLG is not a DAG. For these two SoTs, they can be connected via homo-vendor edge (if they are from the same vendor), TTP-based edge, and vPEP-based edge. Based on Theorem 2, among these edges, only the one with the least weight will finally be picked. That is, instead of incorporating all edges to form an ZT-SLG, for every two SoTs, only the edge with the least weight will be appended. Accordingly, Eq. (23) shows the weight of an edge in ZT-SLG.

\[
\begin{align*}
\forall i, i + 1, \beta, \alpha, n_i, n_{i+1} \in ZT \\Rightarrow \quad w_{i+1,i+1} = \min_{0 \leq \beta \leq \alpha} \left\{ E_{i,i+1}^{\beta} \right\} \quad (23)
\end{align*}
\]

We propose the ZT-SLG generation algorithm as shown in Algorithm 1. To begin, ZSG initializes the node and edge sets as empty (Line 3). For each iteration \( i \), Lines 6 appends the set of physical nodes providing SFI of \( f_i \) to \( N_{ZT} \). It is worth noting that, \( N_{ZT}^0 = S \) (source), while \( N_{ZT}^{\tau+1} = D \) (destination). Lines 7-12 repeatedly calculate the weight of every possible edge between the \( i^{th} \) and \((i-1)^{th}\) layers. Lines 13-14 update the node and edge set in ZT-SLG. Line 17 returns the created ZT-SLG. Fig. 11 shows the ZT-SLG created by ZSG.

With the ZSG technique, our ZAN can be conducted as shown in Algorithm 2. ZAN first initializes ZT-SLG and SFP as empty sets (Line 3). Next, Line 4 calls the ZSG technique to create the ZT-SLG. Then, Line 5 identifies the shortest connecting the source and destination in ZT-SLG. Line 6 checks whether enough resources can be provided to host the shortest path or not. If yes, Line 7 sets the SFP as this shortest path and returns it; otherwise, Line 9 outputs the empty set, which means a rejection. Based on the above process, the SFP will be created as \( A_0 \rightarrow D_1(f_1) \rightarrow C_2(f_2) \rightarrow E_3 \), whereas \((A_0, D_1)\) and \((D_1, C_2)\) employ the homo-vendor verifications, while \((C_2, E_3)\) installs a vPEP. The generated SFP is the same as the optimal result in Fig. 8b with an overall cost of 14.

Based on Theorem 3, if ZAN returns an SFP meaning that sufficient PN resources are provided for this SFC request, then this SFP incurs the least overall cost for deploying such an NSR. Now, we analyze the time complexity of our proposed ZAN algorithm. Again, the ZAN algorithm can be specified into two parts: (i) ZT-SLG generation, and (ii) shortest path identification. For the former part, in the worst case, every node can provide all types of functionalities, and \( |N_{j}^{ZT}| = |N|, j \in [1, \tau] \). In this case, \( |N_{ZT}| = |\tau| * |N| \).
Meanwhile, the number of TTPs is set in the range of \( \{3, 4, 5\} \). When employing the Dijkstra’s algorithm to obtain the shortest path, the time complexity of creating a ZT-SLG is \( O(|V|^2 + |E|) \), whereas \( ZT\)-SLG can be \( O(|V|^2 log |N| + |V|^2 |L| * log |N|) \). Accordingly, the runtime complexity of obtaining a shortest path in ZT-SLG is \( O((|V|^2 |log |N| + |N|^2 |L| * log |N|)) \), which can be further deduced to \( O(|V|^2 |log |N| + |N|^2 |L| * log |N|) \). As a result, the runtime complexity of ZAN is \( O(|N|^2 |log |N| + |N|^2 |L| * log |N|) \).

### VII. Experimental Results and Analysis

#### A. Experimental Settings

We conduct our experiments on NY-20, which is extended from NY-16 [39]. Fig. 12 shows the topology of NY-20, whereas the links in different colors demonstrate network degrees from 2 to 5. Similar to the very state-of-the-arts on SFC deployment, our experiments are also conducted via simulations [40]–[42]. Unless otherwise specified, we generate the parameters as the existing simulation settings [43]. In the physical network, there exists 8 types of SFs and 4 vendors. Meanwhile, the number of TTPs is set in the range of \{1, 2, 3, 4, 5\}, while each vPEP instance needs 0.6 CPU core to be implemented on an SoT. The TTP conversion process will randomly pick an SoT and convert it to a TTP. Each node has the computation capacity of \{4, 8, 12, 16, 20\} CPU cores, and provides \{1, 5\} type(s) of SF instances. Every link has a bandwidth capacity of \{20, 30, 40, 50, 60\} MBps and has the weight in the continuous-discrete range of \[0.1, 0.4\]. In NSRs, the number of required SFs is in the continuous-discrete range of \[2, 6\], each of which demands the computation resources in the range of \[0.1, 0.4\] CPU cores. The bandwidth demand is in the range of \{1, 2, 3, 4, 5\} MBps. The source and destination are randomly picked from the physical network. The computation renting cost is 35 per CPU core per day, while the bandwidth cost is 0.005$ per MBps [44], [45].

#### B. Benchmarks and Performance Metrics

Here, we employ the following schemes as our benchmarks.

**SFC Deployment without Verification (SDw/oV):** (i) employing the technique in [20] to deploy an NSR with the least deployment cost; and (ii) if there exists enough resources to host this SFP, then return the result; otherwise, reject.

**SFC Deployment with Cheapest Verification (SDw/V):** (i) obtaining an SFP from SDw/oV; (ii) for each path hosting a pair of adjacent SFs, employing the cheapest verification method; and (iii) if there exists enough resources to accommodate this SFP, then return the result; otherwise, reject.

**TTP-based SFC Deployment (TTP-SD):** (i) generating a TTP-based SLG and finding the SFP via the Dijkstra’s algorithm; (ii) eliminating the verification cost incurred between every pair of homo-vendor servers along the SFP; and (iii) if there exists enough resources to host such an SFP, then return this SFP; otherwise, reject.

**vPEP-based SFC Deployment (vPEP-SD):** This benchmark is similar to the above one. The only difference is that the step (i) generates the vPEP-based SLG.

**Brutal-Force-based Approach:** (i) generating a layered graph and listing all possible SFPs via Yen’s algorithm [46]; and (ii) incorporating all possible validation to get the optimal result. This method only runs on physical networks with sufficient resources; thus, every SFC will be accommodated.

We evaluate the performances of our proposed schemes using several metrics, including:

**Total Resources Cost (TC):** TC are the total computation and bandwidth cost, which can be calculated via Eqs. (17) and (18). Moreover, when sufficient resources are provided by the physical network, the optimal deployment cost can be obtained from the layered graph technique [20]. Hence, the total resources cost can be the sum of (i) deployment cost and (ii) verification cost (a.k.a. additional cost).

**Occupation Ratio (OR):** OR is calculated as Eq. (24), where \( |NSR_{Method}^{Opt} | \) represents the number of NSRs that is optimally deployed via the method of TTP-SD or vPEP-SD, and \( |NSR| \) is the total number of incoming requests. The occupation ratio can indicate the number of NSRs that are optimally deployed via the TTP-SD, vPEP-SD or ZAN.

\[
OR = \frac{|NSR_{Method}^{Opt}|}{|NSR|} \tag{24}
\]
Additional Cost to Total Cost (ACTC) Ratio: ACTC ratio reflects how much cost is further incurred by verifications, which is calculated via the following equation.

\[
ACTC_{nsr} = \frac{\text{additional cost}_{nsr}}{\text{additional cost}_{nsr} + \text{deployment cost}_{nsr}} \tag{25}
\]

Acceptance Ratio (AcP): This metric reflects the number of NSRs accommodated by different schemes.

Runtime: We use the runtime to evaluate the running time efficiency of each scheme. Note that, all simulations are built by Java over Windows 10 System equipped with Intel i5-11400 2.60 GHz and 16 GB RAM.

C. Performance Bounds Analysis

To begin, we validate the performance bound of the proposed ZAN algorithm, whereas the PN provides sufficient resources. Here, we employ brutal force and SDw/oV as benchmarks. In Fig. 13, the blue-squared-solid, yellow-circled-dashed, and red-triangled-solid curves represent the performances of ZAN, brutal force, and SDw/oV, respectively. Here, we employ \(| \cdot |\) to refer the cardinality of a certain parameter. Notably, in Figs. 13a-13c, the performance of ZAN overlaps with brutal force, which validates its optimality.

Fig. 13a shows the average TC of the above schemes when varying the number of vendors. As one can see, when more vendors exist in the PN, both ZAN and brutal force need higher TC, while SDw/oV remains the same. This shows that the more the vendors are, the higher probability that the required SFs are implemented by different vendors, which incurs additional cost for verification. It is worth noting that, when |vendor| = 1, all schemes have the same performance, which demonstrates the optimality of these three schemes in homo-vendor-only networks.

Fig. 13b shows impact of network degree on the average TC. All schemes require less TC when increasing the network degree. This can be explained as follows. Higher network degrees will (i) reduce the average distance among SoTs statistically, and (ii) increase the number of homo-vendor SoT pairs; thereby, jointly reducing both the deployment cost and additional cost for verification.

Fig. 13c shows the impact of the TTP conversion process on the average TC. Interestingly, SDw/oV needs higher average TC when increasing the number of TTPs. For ZAN and brutal force, the average TC first decreases when |TTP| ≤ 3 and increases when it is higher than 3. This can be explained via the negative and positive impact of the TTP conversion process. We first illustrate the negative impact brought by the TTP conversion process to the deployment cost via the example in Fig. 14. Before the TTP conversion process, the SFP goes through node A (f₁) to node B (f₂), whereas the homo-vendor verification is employed. After the TTP conversion process, node B converts into a TTP and no SF could be host by it. Thus, the SFP has to take the detour of node A (f₁) to node C (f₂) via TTP B for verification. As one can see, one more hop is introduced to the SFP due to the TTP conversion process. The negative impact of the TTP conversion process will reduce the computation and SFI availability of the PN, which likely in turn brings additional deployment cost for embedding an NSR.

Next, we elaborate the positive impact of the TTP conversion process. When a very limited number of TTPs exist (e.g., |TTP| = 1) in the multi-vendor networks, most NSRs have to take longer detour or frequently install vPEP for verifications incurring non-ignorable additional cost. Hence, in this case, more TTPs can reduce such additional cost.

Table IV shows the average performances of employing ZAN, brutal-force-based approach, and SDw/oV. Due to the brutal-force nature, brutal-force-based approach needs a very high runtime compared to both ZAN and SDw/oV. Since three types of edges need to be created by ZAN, our ZAN algorithm needs higher runtime than SDw/oV.

**TABLE IV:** Average performance of the above schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Brutal Force</th>
<th>ZAN</th>
<th>SDw/oV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime (seconds)</td>
<td>1412.254</td>
<td>1.671</td>
<td>0.352</td>
</tr>
<tr>
<td>TC</td>
<td>11.449</td>
<td>11.449</td>
<td>9.653</td>
</tr>
</tbody>
</table>

Fig. 13: Performance bounds analysis of ZAN.
Overall, from the above results, we find that (i) ZAN achieves the optimality as the brutal-force-based approach does and keeps a relatively good runtime-efficiency; (ii) the TTP conversion process has both positive and negative impact on the overall cost to deploy an NSR; and (iii) the verification process will introduce an average of 18.61% additional cost compared to the optimal deployment cost (i.e., SDw/oV).

### D. Physical Networks with Sufficient Resources

To further investigate the performance of ZAN, we also compare ZAN with TTP-SD, vPEP-SD, and SDw/V in the PN with sufficient resources. In Figs. 15 and 16, the average $TC$ of ZAN, SDw/V, TTP-SD, and vPEP-SD are denoted by blue, green, red, and yellow bars in Figs. 15a and 16a. Notably, the solid and dotted bars represent the routing and computation cost, respectively. The average $OR$ is shown in Figs. 15b and 16b, whereas the green, red, and blue bars represent the optimal results that only employ the TTP-based, vPEP-based, and hybrid verifications, respectively. Figs. 15c and 16c show the ACTC ratio of ZAN, SDw/V, TTP-SD, and vPEP-SD, which are denoted by blue-squared, green-circled, red-triangled, and yellow-diamonded curves, respectively.

As shown by Fig. 15a, the average $TC$ of all schemes increases with the SFC length. As one can see, TTP-SD requires the highest routing cost while the vPEP-SD needs the highest computation cost based on their verification features. Meanwhile, even though both ZAN and SDw/V can flexibly employ all types of verifications, SDw/V has a worse performance than that of ZAN due to its local optimality. Notably, when the SFC is short (e.g., $|SFC| = 2$), TTP has a similar performance as ZAN does. When further enhancing the SFC length (from 2 to 6), the average $TC$ increases sharply. This is because, the longer the SFC is, the wider the SFIs might be distributed, whereas more detours are needed for TTP verifications. When $|SFC| = 2$, Fig. 15b shows that more than 50% NSRs can be optimally accommodated via TTP-based verification, while only less than 20% NSRs can be optimally handled by TTP-based verifications. For TTP-SD and vPEP-SD, the ACTC ratio increases with the SFC length as shown in 15c. This shows that the average $TC$ of these two schemes mainly comes from the additional cost incurred by verifications. As ZAN can employ all types of verifications and effectively identify proper verification methods, it requires less additional cost while maintaining a moderate deployment cost, leading to a lower ACTC ratio based on Eq. (25).

Fig. 16 shows the impact of varying the number of TTPs. Similar to our findings above, in Fig. 16a, the TTP conversion process will bring positive (from 1 to 3) and negative (from 3 to 5) impact on the average $TC$. This can also be validated by Fig. 16c, whereas the ACTC ratio of ZAN (SDw/V) decreases when $|TTP| \in [1, 3]$ ($[1, 2]$) and increases when further increasing $|TTP|$. Notably, the positive impact of the TTP conversion process is always stronger than its negative impact for TTP-SD as its average $TC$ decreases with an increasing number of TTPs. This can also be validated by Fig. 16b, whereas only less than 20% NSRs are optimally handled by TTP-based verification when $|TTP| = 1$, and this value increases to almost 60% when $|TTP| = 5$. Fig. 16c also validates this as the ACTC ratio always decreases for TTP-SD. As more SoTs are converted into TTPs, the CPU and SFI availability decreases across the PN; thus, vPEP likely use more computation and routing cost to handle the verifications.

Overall, from our results above, we find that (i) ZAN achieves the best performances regarding the average $TC$; (ii) numerically, ZAN outperforms SDw/V, TTP-SD, and vPEP by an average of 13.06%, 13.36%, and 26.21%, respectively; (iii) due to the local optimality, SDw/V incurs more additional cost than ZAN as shown by Figs. 15c and 16c; and (iv) the ACTC ratio can effectively reflect which impact from the TTP conversion process is stronger.
E. Physical Networks with Limited Resources

We also evaluate the performances of our schemes in networks with limited resources. Note that, 100 NSRs are generated with an even distribution SFC length from 2 to 6. Here, the average \( TC \) only includes the NSRs that are successfully deployed over the PN. In the following figures, the performances of ZAN, SDw/V, TTP-SD, and vPEP-SD are denoted by blue-squared, green-circled, red-triangled, and yellow-diamonded curves, respectively.

![Figure 17: Impact from varying CPU availability.](image)

(a) CPU vs. average \( TC \).

(b) CPU vs. \( AcP \).

Fig. 17: Impact from varying CPU availability.

Fig. 17 shows the impact of varying computation availability when bandwidth resources are sufficient. The increasing amount of computation availability can benefit the performances of all schemes as shown in Figs. 17a and 17b. Among all schemes, ZAN has the best performance in both average \( TC \) and \( AcP \). Notably, when computation availability is 4 and 8, vPEP-SD needs a very high \( TC \) leading to the lowest \( AcP \). This is because the computation availability will run out soon for installing vPEP. For SDw/V, from the \( TC \) and \( AcP \), the violated connections are handled via TTP-based verifications when computation availability is low. It is worth noting that, even with sufficient bandwidth resources, the acceptance ratio of all schemes are under 90%. This is because, when a set of physical nodes that can support a specific type of SF runs out the computation resources (e.g., \( f_1 \) is provided by 5 nodes and all these nodes may have run out of computation resources), the subsequent NSRs might all be denied due to lacking specific SFI resources.

![Figure 18: Impact from varying BW availability.](image)

(a) BW vs. average \( TC \).

(b) BW vs. \( AcP \).

Fig. 18: Impact from varying BW availability.

Fig. 18 shows the impact of varying bandwidth availability when sufficient computation resources are provided. In Figs. 18a and 18b, the increasing amount of bandwidth resources benefit the performance of all schemes. Interestingly, when \( \text{bandwidth} \) is in the range of [20, 60], vPEP-SD outperforms TTP-SD. This can be explained as follows. With a relatively low bandwidth availability, the links around one TTP will run out of bandwidth resources soon, and the subsequent NSRs will have to be detoured to a distant TTP. Since TTP-SD only employs the TTP-based verifications, it requires the highest routing cost of detours to deploy NSRs. The further increasing bandwidth availability mitigates this trend, and the performance of TTP-SD becomes better. Note that, the acceptance ratio of all schemes achieve more than 90% when \( \text{bandwidth} = 60 \). This shows that, when enforcing the ZTaaF paradigms in our experiments, the computation availability plays a more important role than the bandwidth availability as it will impact both SFI installation and vPEP implementation.

Fig. 19 shows the impact from the number of TTPs when limited computation and bandwidth resources are provided. In Fig. 19a, the TTP conversion process brings both positive and negative impact to ZAN and SDw/V. When the number of TTP is no more than 3, the positive impact is stronger than the negative one; otherwise, the negative one is stronger. Meanwhile, regarding the average \( TC \), the TTP conversion process brings stronger positive impact to TTP-SD, while the negative impact is always stronger for vPEP-SD. However, the negative impact of the TTP conversion process is non-negligible on computation and SFI availability. This can be seen from Fig. 19b, even for TTP-SD, the acceptance ratio decreases when the number of TTPs exceeds 3.

![Figure 19: Impact from varying # of TTPs.](image)

(a) # of TTPs vs. average \( TC \).

(b) # of TTPs vs. \( AcP \).

Fig. 19: Impact from varying # of TTPs.

Overall, from our results, we find that (i) computation availability plays a more significant role in SFCD with ZTaaF paradigms; (ii) in terms of \( TC \), ZAN outperforms SDw/V, TTP-SD, vPEP-SD by an average of 10.92%, 21.23%, and 20.49%, respectively; and (iii) the average acceptance ratio of ZAN, SDw/V, TTP-SD, vPEP-SD is 86.56%, 84.21%, 80.29%, and 76.55%, respectively.

VIII. Conclusion

In this work, we have comprehensively studied the cost optimization in SFCD with ZT secure architectures. To enhance the verification flexibility, we introduced the concept of ZT as a function (ZTaaF). Under the ZTaaF paradigms, we have mathematically modelled a problem called SFCD with ZTaaF (SFCZT) with the objective of minimizing the overall cost. We have analyzed SFC-ZT via thorough mathematical proofs and accordingly proposed an exact solution called ZTaaF-aware SFC embedding (ZAN) for networks with...
sufficient resources to handle the incoming request. Extensive simulations have shown that (i) the proposed ZAN achieves optimality within a comparable runtime when given sufficient network resources; (ii) the verification methods in ZTaaS paradigms will introduce an average of 18.61% additional cost; (iii) converting SoTs into TTPs will bring positive or negative impact on ZAN’s performance; and (iv) ZAN outperforms all benchmarks regarding the average total cost and acceptance ratio. In the future, one can study how to effectively distribute the TTPs in a PN when only a limited number of TTPs are available. Another interesting issue is how to cost-effectively deploy SFCs when diverse configurations of SF/vPEP are available in multi-vendor networks. One last question pertains to how much security might be sacrificed by using vPEPs compared to hardware-based PEPs in pursuit of flexibility.

REFERENCES


Danyang Zheng received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2016, and the Ph.D. degree in computer science from the Georgia State University, Atlanta, GA, USA, in 2021. He is currently an Associate Professor at Southwest Jiaotong University, China. His research interests include network function virtualization, network reliability and security, in-network computation, and combinational optimization. He is on the editorial board of Big Data Mining and Analytics. He was Track chair of WCCCT 2024, and served as TPC member of ICC 2021-2023, IEEE ICCC 2023, and IEEE ICCC 2023.

Huanlai Xing received the Ph.D. degree in computer science from the University of Nottingham (Supervisor: Dr Rong Qu), Nottingham, U.K., in 2013. During 2020–2021, he was a Visiting Scholar of computer science, The University of Rhode Island (Supervisor: Dr. Haibo He), Kingston, RI, USA. He is currently with the School of Computing and Artificial Intelligence, Southwest Jiaotong University (SWJTU), Chengdu, China, and Tangshan Institute, SWJTU. He was on Editorial Board of Science China Information Sciences. He was a Member of several international conference program and senior program committees, such as ECML-PKDD, MobiMedia, ISCTT, ICC, TrustCom, IJCNN, and ICINIC. His research interests include semantic communication, representation learning, data mining, reinforcement learning, machine learning, network function virtualization, and software defined networking.

Xiaojun Cao received the B.S. degree from Tsinghua University, Beijing, China, in 1996, the M.S. degree from the Chinese Academy of Sciences, Beijing, in 1999, and the Ph.D. degree in computer science and engineering from the State University of New York at Buffalo, Buffalo, NY, USA, in June 2004. He is currently a Full Professor with the Computer Science Department, Georgia State University, Atlanta, GA, USA. His research has been sponsored by the U.S. National Science Foundation, Centers for Disease Control and Prevention, IBM, and Cisco’s University Research Program. He and his group are working on modeling, analysis, protocols/algorithms design, as well as data processing for networks, and cyber-physical systems. He is a recipient of the NSF CAREER Award from 2006 to 2011. He is a Distinguished Lecturer of IEEE ComSoc from 2019 to 2021 and served as the Chair of IEEE ComSoc ONTC.

Jie Xu is a chair professor of computing with the University of Leeds, the leader for a research peak of excellence with Leeds, director with U.K. EPSRC WRG e-Science Centre, and executive board member with U.K. Computing Research Committee (UKCRC). He has worked in the field of dependable distributed computing for over 30 years. He is a steering/executive committee member for numerous IEEE conferences including SRDS, ISORC, HASE, SOSE and is a co-founder for IEEE IC2E, DAPPS, ICC, etc. He has led or co-led many research projects to the value of over $ 50 M, and published in excess of 400 academic papers, book chapters and edited books. He is the co-founder of two university’s spin-outs that specialize in data analytics and AI software for maximising data centre performance and in co-simulation and digital twins. He is an Alan Turing fellow.