Service-Based Resilience via Shared Protection in Mission-Critical Embedded Networks

Doğanalp Ergenç, *Graduate Student Member, IEEE*, Jacek Rak, *Senior Member, IEEE*, and Mathias Fischer, *Member, IEEE*

***Abstract*—Mission-critical networks, which for example can be found in autonomous cars and avionics, are complex systems with a multitude of interconnected embedded nodes and various service demands. Their resilience against failures and attacks is a crucial property and has to be already considered in their design phase. In this paper, we introduce a novel approach for optimal joint service allocation and routing, leveraging virtualized embedded devices and shared backup capacity for the fault-tolerant design of mission-critical networks. This approach operates in phases utilizing multiple optimization models. Furthermore, we propose a new heuristic that ensures resource efficiency and fault-tolerance against single node and link failures as pre-requisite for resilience. Our experiments for different application scenarios indicate that our heuristic achieves results close to the optimum and provides 50% of capacity gain compared to a dedicated capacity protection scheme. Moreover, our heuristic ensures fault-tolerance against at least 90% of all potential single node failures.**

***Index Terms*—Mission-critical networks, embedded, resilience, shared protection.**

I. Introduction

MISSION-CRITICAL embedded systems as used in autonomous vehicles, airplanes, and industrial networks have evolved to complex ecosystems. For instance, the latest Tesla autopilot[1] is supported by eight cameras and twelve ultrasonic sensors for high precision and high-quality environmental data. Similarly, together with Industry 4.0 intelligent cyber-physical systems emerged that are composed of a multitude of collaborating embedded devices [1], [2] that run safety-critical services.

Moreover, we currently observe that trends from conventional computer networks, like more powerful devices and virtualization, are widely adopted in the (embedded) IoT domain. McKinsey & Company, for example, considers the virtualization as a key technology to satisfy the latency and reliability requirements of the future autonomous driving [3]. Furthermore, there is ongoing standardization activities such as Automotive Virtual Platform Specification [4] and Future Airborne Capability Environment (FACE) [5] that are preparing the usage of open source virtualization technologies in critical in-vehicle and military systems. As a result and in the future, these systems can host multiple virtualized services on a physical node by maintaining process isolation [6], [7].

From a system design perspective, safety- or mission-critical embedded networks host various potentially interconnected services with specific demands, e.g., certain resource consumption or communication with bounded delay. Virtualization techniques helps to place those services over the physical network and then establish their intercommunication according to their requirements. Besides, as especially the safety-critical services should be protected against any disruption such as attacks or failures, the virtualization enables dynamic failover schemes like migrating/recovering services after node failures [8], [9]. When standard safety concepts like replicating devices and services alone are not sufficient in the presence of sophisticated attackers, such a flexibility in design provides resilience against failures and attacks, maintaining availability with graceful degradation in worst-case or full recovery again.

Eventually, services need to be deployed in the embedded network by considering the capacity and capabilities of nodes and their interconnection. Hence, two degrees of freedom are the result: the service placement on nodes and the routing of data flows in between these nodes. Meanwhile, this configuration should guarantee a certain degree of resilience against the potential malfunctions or threats. To satisfy those requirements, we have modeled the resilient service placement and routing problem addressing single node failures in our preliminary work [10]. Leveraging virtualized embedded devices and virtual services, we have found alternative configurations of the network to reserve required resources for migrating services and flows in case of failures. From this point of view, enabling the dynamic service deployment changes the dimension of the resilient communication by benefiting from the flexible design of up-to-date embedded devices [11]. However, as we also show in [10], the resilient service placement and routing problem is very complex and thus impractical to solve for larger problem sizes.

In this paper, we advance our previous work by utilizing capacity sharing for path protection against single link failures. In comparison to the previous approach that we allocated dedicated backup capacity for traffic demands, we aim to further improve the resource-efficiency by enabling shared capacity use with proper service and demand configurations. We formulate a

Manuscript received November 20, 2020; revised February 17, 2021; accepted February 19, 2021. Date of publication February 26, 2021; date of current version September 9, 2021. The associate editor coordinating the review of this article and approving it for publication was M. Tornatore. *(Corresponding author: Doğanalp Ergenç.)*

Doğanalp Ergenç and Mathias Fischer are with the Department of Informatics, Universität Hamburg, 20146 Hamburg, Germany (e-mail: ergenc@informatik.uni-hamburg.de; mfischer@informatik.uni-hamburg.de).

Jacek Rak is with the Department of Computer Communications, Gdansk University of Technology, 80-233 Gdansk, Poland (e-mail: jrak@pg.edu.pl).

Digital Object Identifier 10.1109/TNSM.2021.3062461

[1]Tesla, https://www.tesla.com/autopilot

TABLE I
COMPARISON OF THE STATE-OF-THE-ART STUDIES

Study	Resource Efficiency	Optimal Routing	Resilience	Inter-service Dependency	Shared Protection
Ergenc et al. [10]	✓	✓	✓	✓	
Espling et al. [12]	✓			✓	
Breitgand et al. [13]	✓				
[14], [15], [16], [17], [18], [19], [20]	✓	✓			
Medard et al. [21], Lee et al. [22]		✓	✓		
Barla et al. [23], Xu et al. [24]	✓	✓	✓		
Beck et al. [25]	✓		✓		
Atallah et al. [26]		✓	✓		
[27], [28], [29]		✓	✓		✓
He et al. [30]			✓		✓

new optimization scheme extending the one in [10] and then dividing it into multiple steps to reduce its complexity for the embedding of complex inter-connected services by meeting Quality of Service (QoS) and robustness requirements. While our previous model can find the optimal solution for only small-size network, our new multi-step model can scale better ensuring the same degree of fault-tolerance against node and link failures. Accordingly, we evaluate the new model in a more realistic topology, which reflects the characteristics of an avionic network architecture. The resulting system becomes resilient against failures and attacks when it is coupled with a dynamic function migration mechanism that realizes the configuration found by our optimization scheme. Concerning our contributions, we

- propose three separate optimization models to solve (i) service placement and routing, (ii) allocation of backup paths with shared capacity use against link failures and (iii) a service migration scheme in case of node failures. While our service allocation and routing model finds the optimal working paths, the shared backup capacity model results in up to 70% capacity gain in comparison to reserving dedicated backup capacity.
- implement the column generation method to enhance our previous model and to solve the extended problem for larger topologies and service overlays more effectively. The resulting model provides fault-tolerance against all single node failures.
- design a new heuristic utilizing Steiner trees to promote shared backup capacity use and improve resource efficiency. Our heuristic results in near-optimal results for the shared backup capacity allocation. It provides more than 90% fault-tolerance against single random node failures that can happen on the host nodes.

The rest of the paper is organized as follows. Section II summarizes related work. In Section III, we introduce our service-based resilience model and the optimization models for the resilient service deployment and routing problem under shared backup capacity. Also, we introduce and explain our heuristic in Section IV. Section V presents our experimental setup and results in detail. Lastly, we summarize our solution and findings in Section VI.

II. RELATED WORK

In this section, we shortly summarize the requirements of our problem. Then, we summarize related work on service allocation, network resilience, and lastly, capacity sharing for backup protection schemes.

Table I shows the requirements for optimal resilient embedded network design that we used as criteria to compare all other studies presented in the rest of this section qualitatively. *Resource Efficiency* and *Optimal Routing* represent the optimality in resource and network utilization for service deployment and traffic engineering, respectively. *Resilience* is one of the main concerns to protect networks against failures or attacks and should be considered for an optimal design of a mission-critical network. *Inter-service Dependency* represents the relationship, e.g., hierarchy or communication, between different services since they are interconnected having specific requirements. Lastly, *Shared Protection* is a concept to use network resources more effectively and is important for the networks that should be more compact and low cost, e.g., for a car or airplane to be lighter and cheaper. In the rest of this section, we discuss the related work according to those criteria.

Service Allocation: In the domains of cloud computing, Software-Defined Networking (SDN), and Network Function Virtualization (NFV), a *service* represents a movable (or relocatable) function of a particular type and characteristics that is allocated to physical nodes. In cloud computing, a service generally provides some specific content, an application, or a platform to users under certain Service-Level Agreements (SLAs) and by minimizing the operational costs at the same time. It requires accurate resource orchestration regarding where, when, and how many service instances are deployed [31], [32]. Besides, the dependencies of services on each other [12], service migrations [13], load-balancing [14], and task scheduling [15] affect the costs of providers and also impact the user experience as well.

SDN/NFV services are considered as virtual functions to process and regulate the communication such as firewalls, routers, and load balancers, or provide network-wide services such as Domain Name System (DNS) and authentication, authorization, and accounting (AAA) services. The proper allocation of those services [33], [34] is important to, for instance, minimize operational costs [16] and physical resource fragmentation [17] for the providers, and maximize the service quality [16] and responsiveness [18] for the user experience. Various other studies address the optimum service allocation and routing problem jointly to deploy the

services on the paths [19], [20] to utilize network resources optimally.

Contrary to existing works, the service deployment scheme proposed in this paper focuses on emerging virtualized embedded networks. As the communication traffic is defined between services, inter-service relationships are decisive for network design considering both service deployment and the traffic engineering. Therefore, it is a joint service allocation and inter-service traffic routing problem where routing also depends on the service allocation. Moreover, adding resilience requirements to such a dynamic deployment scheme renders the problem even more challenging.

Network Resilience: Many traditional approaches leverage graph-related properties of networks to increase their robustness. Against link failures, for instance, finding primary and redundant directed trees [21] as well as multiple disjoint paths [22] have been proposed. Some other related studies present the optimization problems with resilience constraints. In [23], the authors optimize virtual cloud topologies having k redundant instances under network constraints. Similarly, [24] creates survivable virtual groups for each service to guarantee their availability and formulate the deployment of the groups to an underlying network as an optimization problem. Both studies focus on cloud service characteristics. In [25], a resource allocation model is proposed for SDN/NFV, including fault-tolerance constraints. The authors of [26] consider topology synthesis, routing, and scheduling problems jointly for fault-tolerance in Time-Sensitive Networks (TSN) without including any resource utilization constraint.

In our preliminary work [10], we have considered the resilience of services together with optimal resource allocation and routing for inter-service communication but only for small random networks, i.e., up to ten nodes and seven services. In comparison to other studies here, our new approach can find both the optimal and resilient communication scheme for mission-critical embedded networks in a more resource-efficient manner.

Shared Backup Protection: Shared backup protection increases the resource efficiency in network design as it enables using an amount of capacity mutually between different flows or demands under certain conditions. It is especially prevalent in optical networks where the backup paths can share wavelength links when their working paths are disjoint [35], [36]. In [27], the authors calculate shared backup paths to protect content-connectivity between users and optical datacenters in case of disasters. They propose an ILP and a heuristic to minimize the total number of spectrum slots for optical connections. Reference [28] proposes an efficient shared protection scheme without increasing the length of backups paths compared to the respective working paths and thus promotes fast service recovery. The author formulates the shared protection problem as an ILP for the networks utilizing wavelength-division multiplexing (WDM) and presents heuristics to solve that NP-complete problem. In [30], the authors leverage capacity sharing for survivable virtual network embedding in optical networks to decrease the rejection ratio of an incoming demand due to the lack of resources. They propose a polynomial-time heuristic to calculate shared backup paths. Lastly, [29] utilizes shared protection for a fault-tolerant topology design against more than one failure optimizing capacity usage with two ILP models. Even though all of those studies offer a degree of resilience against node and link failures via shared protection, they assume only static demands that are predefined between certain nodes. Similar to our discussion for the service allocation problems, one of our primary concerns is the deployment of service instances together with working and backup paths where the allocation of demands depends on such deployment and thus is dynamic. Additionally, we have to ensure the resilience of the service deployment, not only the data traffic.

Consequently, in state of the art, the studies do not focus on the problems of service allocation, routing, and resilience jointly in a resource-efficient manner. Those problems are highly relevant for the design of mission-critical networks, which is an avionic network in our use case shown in Section V-A. In that use case, we reflect the domain-specific aspects regarding the topology and service overlay. Adding our shared capacity scheme on top of that, we aim to reduce the resource use and thus the cost of the system. Accordingly, we propose heuristics that solve those problems altogether. From those perspectives, we believe that we propose a solution to a problem that has not been studied holistically yet. Although in our former work [10] we addressed most of those problems, that approach cannot find optimal solutions for networks of reasonable size.

III. Service-Based Model for Embedded Networks

Our model aims to embed an overlay network of services into an underlying physical network so that the resulting assignment maintains the inter-service data traffic, latency, and foremost resilience demands. In this sense, the service overlay describes a communication scheme between service instances having certain inter-communication demands. It can also implicitly reflect redundancy for a service, e.g., including multiple service instances in a distributed manner. In this section, we propose an optimization scheme to find the optimal embedding of a service overlay to the physical network. First, we give an overview of the model and our optimization approach that consists of three optimization phases. Then, we explain each phase in more detail.

A. Overview of the Model and the Optimization Scheme

In the service-based model, a service $s \in S$ is defined as a function or virtual instance to be deployed on a physical node $v \in V$. Each *S*has certain resource requirements τ_s, e.g., CPU or memory, and different criticality levels, e.g., mission-critical or best-effort. Those levels impose a deployment constraint in which only particular nodes can host the service instances with certain criticality, i.e., $k_{sv} = 1$. Another important term, demand $d \in D$, specifies inter-service communication requirements in terms of the end-to-end latency l_d and the amount of data traffic h_d to be exchanged, e.g., required bandwidth. That is, a demand is defined between two service instances and it conditions the data communication in between.

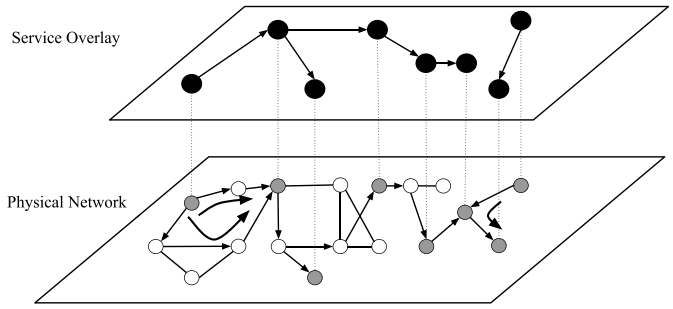

Fig. 1. Service overlay on top of the underlay physical network. Dashed lines show how basic service instances are assigned to physical nodes. Grey nodes host the service instances and directed edges show the paths that carry the traffic demands.

Fig. 1 gives an example for the embedding of a service overlay (black nodes) in the underlying physical network (grey nodes). While a link between two services (an edge between two black nodes) represents a demand, the connection of two physical nodes (an edge between two grey nodes) is a physical link $e \in E$, i.e., having a nominal bandwidth capacity, in the network. A service instance can be allocated on a single node to establish communication with other nodes that host neighboring service instances. The overall deployment should be restricted by (i) the node resource capacities, i.e., r_v for node v, consumed by the services and (ii) link capacities, i.e., c_e for link e, required by the inter-service demands. Besides, the delay induced by path $p \in P$ between two nodes hosting the communicating services imposes a further restriction on the latency on the respective demand.

As shown in [10], a single optimization model that reflects optimal service deployment, optimal routing configuration, and fault-tolerance under different failure scenarios results in high complexity. Thus, even for small networks and few services, it might take up to several days to find a configuration with minimum communication latency and a guaranteed resilience against single node failures. For this reason, we split the problem into three phases and different optimization models that need to be solved subsequently as a part of the whole optimization scheme. Those phases that are shown in Fig. 2 are:

1) *Bootstrapping:* In this phase, we find an initial configuration with service deployments and working paths. For that, we formulate an integer linear problem (ILP), namely Bootstrapping-ILP, to find the shortest working paths within the limited node and link resources.
2) *Shared Backup Protection:* We establish shared backup paths against possible link failures on the working paths found in the phase 1. Using the solution of the previous phase as input, we formulate another ILP, namely Backup-ILP, to minimize the use of backup capacity by maximizing shared protection.
3) *Service Migration:* We search for the backup nodes, which communicate with other host nodes with minimum latency, to migrate services in the case of node failures. In this phase, we formulate another optimization model, namely Migration-LP.

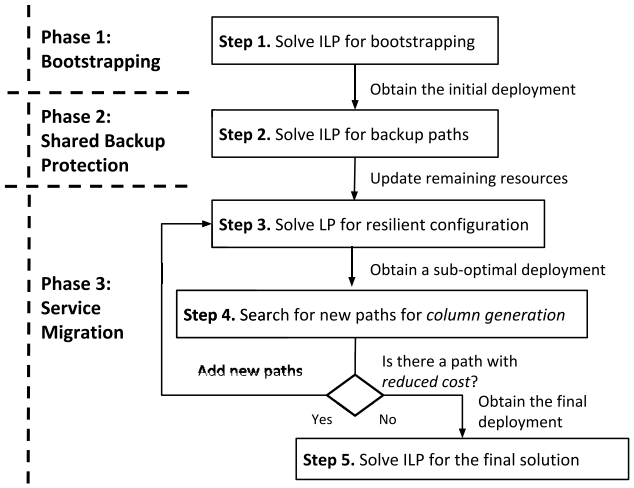

Fig. 2. Multiple steps for the optimal configuration of a resilient virtualized network.

Splitting the model into three phases eases the formulation of the constraints. For instance, finding working and backup paths, in phases 1 and 2, could also be considered as a single phase as they are highly dependent. However, it eventually results in complex non-linear constraints, i.e., having complexity higher than quadratic equations. Apart from avoiding such constraints to a certain extent, we also linearize the remaining non-linear constraints to make the overall optimization scheme easily solvable by the existing linear optimization tools. As we solve those linearized models in our experiments, we here introduce the original models as ILP and LP omitting further linearization details. For that, we utilize the McCormick envelopes [37] introducing extra variables and constraints, whose details and complexity are extensively discussed in our preliminary study [10].

Note that splitting the problem into the different optimization phases results in individual optimal solutions for each phase, not a global optimum for the whole scheme involving all constraints at once. We consider this reduction as a trade-off to get a solution which is closer to the optimal one for larger problem instances. Nevertheless, computing the approximation ratio of the split model to a possible singular model requires to formulate and solve such a complex model, which is not practical as we discussed in [10].

In the following subsections, the respective optimization models are presented. Table II briefly summarizes all terms and definitions used in those optimization models.

B. Bootstrapping

In the bootstrapping phase, we solve the optimization model, Bootstrapping-ILP, to find the initial configuration where the service instances and traffic demands are placed on the nodes and working paths. This phase constitutes a base configuration to build further reconfigurations, i.e., service and flow migrations, in the case of node failures and to find backup paths against link failures. The description of all relevant parameters is given in Table II.

TABLE II
TERMS AND DEFINITIONS IN THE OPTIMIZATION PROBLEM. *Base* TYPE CONTAINS THE FUNDAMENTAL ELEMENTS OF THE OPTIMIZATION SCHEME. *Constant*S ARE NETWORK- AND SERVICE-RELATED PARAMETERS GIVEN AS INPUT. *Variable*s REPRESENT THE PARAMETERS TO BE OPTIMIZED

Type	Symbols	Set	Interval	Definition
Base	u, v	V	-	Nodes in the network
	e	E	-	Link (edges) between nodes
	s, t	S	-	Basic services
	d, q	D	-	A demand between a pair of services
	q	H_d	-	A demand that can share backup capacity with demand d
	p	P_{uv}	-	A path between nodes u and v
	p	$P_{\bar{d}}$	-	A backup path which is disjoint to the working path of d
	f	F	-	A failure scenario
Constants	τ_s	$\Re^*$	[0,∞]	Resource consumption of s
	h_d	$\Re^*$	[0,∞]	Traffic volume of d
	c_e	$\Re^*$	[0,∞]	Maximum link capacity of e
	c_e^*	$\Re^*$	[0,c_e]	Reduced link capacity of e
	r_v	$\Re^*$	[0,∞]	Maximum resource capacity of v
	r_v^*	$\Re^*$	[0,r_v]	Reduced resource capacity of v
	l_d	$\Re^*$	[0,∞]	Latency requirement of d
	l_e^*	$\Re^*$	[0,∞]	Latency in e
	k_{sv}	Z^*	[0,1]	Indicates if v is capable to host s
	a_{vf}	Z^*	[0,1]	Indicates if v is available in failure state f
	θ_{pf}	Z^*	[0,1]	Indicates if p is available in failure state f
	α_{pe}	Z^*	[0,1]	Indicates if p includes e
	x_{dp0}	Z^*	[0,1]	Indicates if a flow is allocated to path p of demand d in a non-failure scenario ($f = 0$)
	y_{sv0}	Z^*	[0,1]	Indicates if s is hosted by v in a non-failure scenario ($f = 0$)
Variables	x_{dp}	Z^*	[0,1]	Decides if d is assigned to p
	x_{dpf}	Z^*	[0,1]	Decides if d is assigned to p in scenario f
	y_{sv}	Z^*	[0,1]	Decides if s is hosted by v
	y_{svf}	Z^*	[0,1]	Decides if s is hosted by v in scenario f
	z_{dp}	Z^*	[0,1]	Decides if d is assigned to backup path p
	z_{de}^*	Z^*	[0,1]	Decides if d is assigned to e
	g_{de}	Z^*	[0,1]	Decides if any demand $q \in H_d$ is assigned to e

Bootstrapping-ILP is given below. x_{dp} and y_{sv} are two binary decision variables to indicate if demand d is assigned to path p and if service s is deployed on node v, respectively.

$$\min \sum_{d \in D} \sum_{p \in P} x_{dp}|p| \tag{1}$$

$$\sum_{s \in S} y_{sv}\tau_s \leq r_v \quad \forall v \in V \tag{2}$$

$$\sum_{v \in V} k_{sv}y_{sv} = 1 \quad \forall s \in S \tag{3}$$

$$x_{dp} \leq y_{sv}y_{tu} + y_{tv}y_{su} \quad \forall d \in D, \forall u, v \in V, \forall p \in P_{uv}, (s,t) \in d \tag{4}$$

$$\sum_{d \in D} \sum_{\substack{p \in P, \\ e \in p}} x_{dp}h_d \leq c_e \quad \forall e \in E \tag{5}$$

$$\sum_{e \in p} x_{dp}l_e^* \leq l_d \quad \forall d \in D, \forall p \in P \tag{6}$$

$$\sum_{p \in P} x_{dp} = 1 \quad \forall d \in D \tag{7}$$

The objective function (1) minimizes the length of selected paths, where $|p|$ represents the path length. Minimizing the total path length can be considered as both performance and cost optimization. That is, allocating shorter paths enables establishing low-latency communications, i.e., here with less hops, and decreasing the number of occupied links, which is especially important for mission-critical networks to reduce the cost and the complexity of the system.

Constraint (2) and (3) ensure that *v*has sufficient resources to host *s* and *s* is deployed on exactly one node that is capable to host *s* (e.g., equipped with the required hardware). Constraint (4) restricts the flow assignment in a way that *d* can be deployed on *p* if only the source and destination nodes *u*, *v* of *p* host required services *s* and *t*. Constraint (5) ensures that each link *e* of *p* has sufficient resources, e.g., bandwidth, to carry the traffic of *d* if it is assigned to *p*. While constraint (6) ensures that *p* is selected to satisfy the maximum tolerable latency for *d*, constraint (7) guarantees that *d* is assigned exactly to one working path. Note that, as inferred in the latest constraint, traffic demands are assumed to be non-bifurcated.

There are two significant uses of the bootstrapping for the next two phases. In the shared protection phase, the bootstrapping eases finding the optimal shared backup paths by providing an initial configuration to (i) detect disjoint working paths that can share backup capacity and (ii) select disjoint backup paths for the given working paths. Formulating mutual disjoint paths, i.e., a working path which is disjoint to both its backup path and other working paths to leverage shared capacity, together with the service allocation results in a complex optimization model (e.g., having cubic constraints). Therefore, computing working paths in advance and then formulating the shared backup protection problem is more convenient to be solved by existing optimization tools. In the service migration phase, the bootstrapping provides a basis of nodes, i.e., host nodes, whose failures are disruptive and result in the unavailability of service instances. Besides, it enables us to keep a part of the initial deployment and to avoid the migration of services and flows on non-failed nodes. Therefore, the bootstrapping

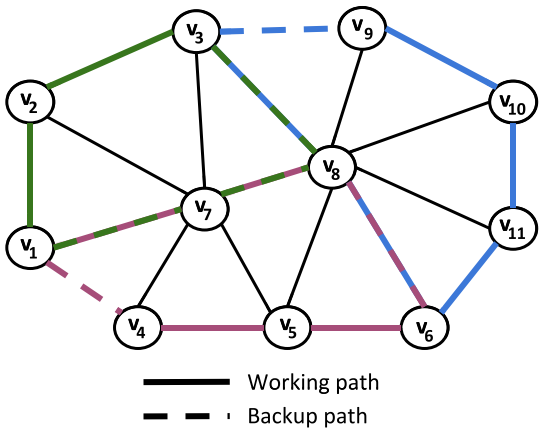

Fig. 3. Example scenario of sharing the backup path capacity.

phase reduces the search space of the optimization problem, i.e., the number of node failures to consider and the services and flows to reconfigure.

Lastly, in Bootstrapping-ILP, as constraint (4) conforms to both service allocation and routing restrictions, it increases the complexity of the model in $\mathcal{O}(|D||P||S|^2|V|^2)$ in terms of the number constraints. However, practically, only a limited number of such constraints are effective as (i) only the respective services and (ii) only the paths are considered for each demand and each pair of nodes, respectively. Moreover, as it is also a non-linear constraint, the linearization of the multiplication of two binary variables adds extra constraints as well. In terms of variables, due to x_{dp} and y_{sv}, Bootstrapping-ILP is bounded by $\mathcal{O}(|D||P| + |S||V|)$ variables.

C. Shared Backup Protection

In this phase, we formulate another ILP, Backup-ILP, to find backup paths for each demand. Providing communication resilience using dedicated backup paths is generally costly. In particular, if 100% of requested throughput has to be available after a failure, the amount of resources (link capacities) needed to set up the backup path is higher than the resources needed for establishing the working path. To improve the resource efficiency, the concept of sharing the link capacities assigned to backup paths can be applied in such scenarios [38]. In general, sharing the link capacity among several backup paths at a given link is possible if these backup paths protect against different failure scenarios as illustrated in Fig. 3. In the figure, three demands are assigned to the working paths v_1-v_2-v_3, v_4-v_5-v_6, and v_9-v_{10}-v_{11}-v_6, and the backup paths v_1-v_7-v_8-v_3, v_4-v_1-v_7-v_8-v_6, and v_9-v_3-v_8-v_6, respectively. Accordingly, the shared backup capacities are reserved at links v_1-v_7 and v_7-v_8 by the first and second backup paths, at link v_3-v_8 by the first and third backup paths, and at link v_6-v_8 by the second and third backup paths for the case of protection against a single node failure as the respective working paths are disjoint and thus subject to different scenarios of single failures. In particular, concerning the scenario of a single link (or a single node) failure covering the majority of failure cases [39], backup path sharing is possible if the respective working paths are mutually link-disjoint (or node-disjoint) [28], [40].

As presented in detail in [28], for a demand $d$with the requested throughput h_d, the respective backup path at link $e$in the case of shared protection would require the allocation of (i) no extra capacity if the amount of shareable capacity c^+ already allocated to backup paths at link $e$is at least h_d or (ii) the extra capacity of $h_d - c^+$ in all other cases. Here, the shareable capacity can be considered as the capacity already reserved for a backup path of another demand q that is accepted earlier and not affected by the same link failure affecting a working path of d. In case $h_q < h_d$, extra capacity $h_d - h_q$ needs to be reserved at the link even though h_q amount of capacity can be used by both q and d in case of (different) link failures.

Our model Backup-ILP is given below. Before formulating the problem for a topology G, we update link capacities c_e to c_e^* to denote the capacity not allocated for working paths. Using the initial configuration (i.e., working paths) as the input, we construct a set H_d for each d. It includes demands $\{q_1, q_2 \ldots\}$ that (i) induce shareable backup capacity with demand $d$as they have disjoint working paths with $d$and (ii) have larger traffic demands $h_q > h_d$. As a result, if $q \in H_d$ is assigned on link e, h_d is not necessary to consume extra capacity. The binary decision variables z_{dp} and z_{de}^* represent whether demand $d$is assigned to path $p \in P_{\bar{d}}$ and link e, respectively. Here, $P_{\bar{d}}$ is a set of disjoint paths to the working path of d obtained from the previous phase and it is computed beforehand. The other decision variable g_{de} shows if any $q \in H_d$ is already assigned to link e. g_{de}^- is the negation of g_{de}.

$$\min \sum_{e \in E} \sum_{d \in D} z_{de}^* g_{de}^- h_d \tag{8}$$

$$\sum_{p \in P_{\bar{d}}} z_{dp} = 1 \quad \forall d \in D \tag{9}$$

$$z_{de}^* \geq z_{dp} \quad \forall d \in D, \forall e \in E, p \in P_{\bar{d}} \wedge e \in p \tag{10}$$

$$z_{de}^* \leq \sum_{\substack{p \in P_{\bar{d}}, \\ e \in p}} z_{dp} \quad \forall d \in D, \forall e \in E \tag{11}$$

$$g_{de} \geq z_{qe}^* \quad \forall d, q \in D, \forall m \in H_d, \forall e \in E \tag{12}$$

$$g_{de} \leq \sum_{q \in H_d} z_{qe}^* \quad \forall d \in D, \forall e \in E \tag{13}$$

$$\sum_{d \in D} z_{de}^* g_{de}^- h_d \leq c_e^* \quad \forall e \in E \tag{14}$$

In the model, the objective function (8) minimizes the total shared backup capacity. It aims to increase the resource-efficiency and eventually decrease the design cost of the system by occupying less backup resources.

Constraint (9) ensures exactly one backup path assigned for each demand d. Constraints (10) and (11) configure z_{de}^* for each link $e$checking if p involving $e$is a backup path for demand d, i.e., $z_{dp} = 1$. Similarly, constraints (12) and (13) configure g_{de} checking if any demand $q \in H_d$ is assigned

to link e. Lastly, constraint (14) ensures that the required resources for $d$with the highest traffic demand are reserved.

As an exception, if demand d is not suitable to use any amount of shared capacity with another demand, then $H_d = \{\}$. In this case, $g_{de} = 0$ holds strictly and the full amount of traffic for demand $d$should be assigned to the respective links without considering any shared capacity. Constraint (14) implicitly considers this scenario as well.

Lastly, in terms of complexity, Backup-ILP is bounded by $\mathcal{O}(|D||E||P|)$ constraints and $\mathcal{O}(|D|(|E| + |P|))$ variables including linearized the non-linear constraint, which contains the multiplication of z^*_{de} and g_{de}.

D. Service Migration

Backup paths protect traffic demands against the failures in the intermediate nodes and links of the respective working paths. However, the failure of one of the end nodes hosting a service still disrupts the services and demands. In the last phase, we formulate Migration-LP to find alternative nodes for each service hosted by failed nodes. Defining failure scenarios $f \in F/\{0\}$, we consider the failure case of each node that hosts a service according to the given bootstrapping configuration.

Before formulating the problem for the topology G, we update node and link capacities according to the deployments in the previous phases. In Migration-LP, x_{dpf} and y_{svf} are the binary decision variables that represent whether demand d is assigned on path p and if service $S$is deployed on node v in the failure scenario f, respectively. Each scenario f is represented by a vector of binary variables a_{vf} that specficies whether node $v$is not failed in scenario f. Services can be hosted at node $v$only if $a_{vf} = 1$. θ_{pf} is another binary variable that represents if path p is not broken, i.e., is usable, in scenario $f$and is decided by the availability of the nodes on p s.t $\theta_{pf} = \prod_{\substack{v \in V, \\ v \in p}} a_{vf}$. According to the service configuration from the bootstrapping phase, the failure scenarios are defined in such a way that each one represents the failure of a single host node, i.e., $\sum_{v \in V} a_{vf} = 1$. Therefore, the number of scenarios $|F|$ equals to the number of distinct nodes hosting services in the initial configuration. Eventually, the resulting configuration of Migration-LP gives an alternative deployment to update the bootstrapping configuration in the case of the respective failure scenarios.

Migration-LP is given below. The objective function (15) minimizes the length of selected paths. Constraints (16)-(21) resemble to constraints (2)-(7) in the bootstrapping phase. Constraints (22) and (23) ensure that if the initial flow assignment and service deployment are not affected by the failure in scenario f, their configuration is kept to avoid the unnecessary reconfiguration of the network. Here, x_{dp0} and y_{sv0} are given as input according to the initial configuration from the bootstrapping phase.

$$\min \sum_{d \in D} \sum_{p \in P} \sum_{f \in F} x_{dpf} |p| \tag{15}$$

$$\sum_{s \in S} y_{svf} \tau_s \leq r^*_v \quad \forall v \in V, \forall f \in F \tag{16}$$

$$\sum_{v \in V} k_{sv} y_{svf} a_{vf} = 1 \quad \forall s \in S, \forall f \in F \tag{17}$$

$$x_{dpf} \leq \theta_{pf} \left[y_{suf} y_{tvf} + y_{tvf} y_{suf} \right] \quad \forall d \in D, \forall u, v \in V, \forall f \in F \quad \forall p \in P_{uv}, (s,t) \in d \tag{18}$$

$$\sum_{d \in D} \sum_{p \in P} x_{dpf} \theta_{pf} \alpha_{pe} h_d \leq c_e \quad \forall e \in E, \forall f \in F \tag{19}$$

$$\sum_{e \in E} x_{dpf} \alpha_{pe} l^*_e \leq l_d \quad \forall d \in D, \forall p \in P, \forall f \in F \tag{20}$$

$$\sum_{p \in P} x_{dpf} \theta_{pf} = 1 \quad \forall d \in D, \forall f \in F \tag{21}$$

$$x_{dpf} \geq \theta_{pf} x_{dp0} \quad \forall d \in D, \forall p \in P, \forall f \in F \tag{22}$$

$$y_{svf} \geq \theta_{pf} y_{sv0} \quad \forall s \in S, \forall v \in V, \forall f \in F \tag{23}$$

Finding the optimal solution of the service migration problem is highly complex mostly due to the introduction of multiple failure scenarios, and it can not be easily found in a reasonable time even for small network instances. Therefore, we apply the column generation method in Migration-LP, adding the candidate paths iteratively to reduce the initial number of variables and constraints to be considered. To be able to apply the method, we use the linear relaxation of binary variables and thus solve the problem as an LP (rather than an ILP).

After solving an initial instance of the service migration problem with a limited set of paths, including working and backup paths, which are found beforehand, we add a new set of candidate paths that possibly improve the objective value. This process is shown as Step 3 and 4 in Fig. 2. To select a candidate path p^*, we define the reduced cost function (24) derived from the Langrangian function (25) of the model using the dual variables of LP.[2] As the paths with positive reduced cost can contribute to the existing feasible solution taken from Step 3, those are added to the used set of paths. Then, the LP is re-solved with the extended set of paths.

$$c_{p^*f} = -\left(\sum_{d \in D} \sum_{e \in E} \alpha_{p^*e} \nu_{ef} \theta_{p^*f} h_d - |p^*| \right) \tag{24}$$

$$\begin{aligned} \mathcal{L}^* \Big(& x_{dpf}, \eta^{uv}_{dpf}, \nu_{ef}, \rho_{dpf}, \pi_{dpf}, \phi_e \Big) \\ &= - \sum_{p \in P} \sum_{d \in D} \sum_{f \in F} x_{dpf} |p| \\ &\quad + \sum_{u,v \in V} \sum_{d \in D} \sum_{p \in P} \sum_{f \in F} \eta^{uv}_{dpf} \left(x_{dpf} - y_{svf} y_{tuf} \theta_{pf} \right) \\ &\quad + \sum_{e \in E} \sum_{f \in F} \nu_{ef} \left(\sum_{d \in D} \sum_{p \in P} x_{dpf} \theta_{pf} \alpha_{pe} h_d - c_e \right) \\ &\quad + \sum_{d \in D} \sum_{p \in P} \sum_{f \in F} \rho_{dpf} \left(\sum_{e \in E} x_{dpf} \alpha_{pe} l^*_e - l_d \right) \\ &\quad - \sum_{d \in D} \sum_{p \in P} \sum_{f \in F} \pi_{dpf} \left(x_{dpf} - \theta_{pf} x_{dp0} \right) \end{aligned} \tag{25}$$

[2]Further details for the column generated method, dual variables, and cost functions can be found in many studies such as [41]–[43].

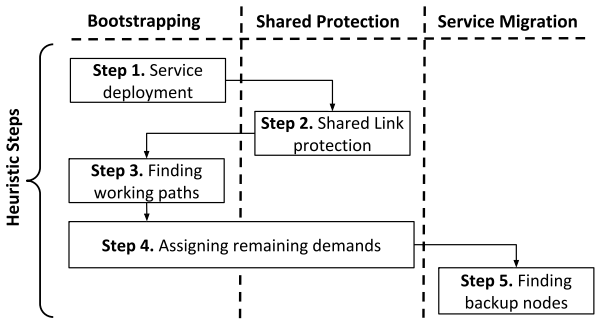

Fig. 4. Correspondence between the steps of the heuristic and the optimization phases.

Lastly, as Step 5 in Fig. 2, if there is no path left to include, e.g., all remaining paths have a non-positive reduced cost, the reduced problem is solved as an ILP (i.e., without linear relaxation) using only the obtained set of all useful paths.

In terms of complexity, Migration-LP has the highest number of constraints and variables in comparison to Bootstrapping-ILP and Backup-ILP. Solving a very similar problem with Bootstrapping-ILP for multiple failure scenarios, it is bounded by $\mathcal{O}(|D||P||F||S|^2|V|^2)$ constraints and $\mathcal{O}(|F|(|D||P|+|S||V|))$ variables.

IV. Heuristic

As described in Section III, three phases constitute the essence of the problem: (i) Service deployment satisfying demands, (ii) shared link protection scheme minimizing the use of link resources, and (iii) service migration scheme for node protection. We propose a 5-step heuristic covering those three phases referring to our optimization scheme. Fig. 4 shows the corresponding steps of the heuristic addressing the same objective as the optimization models. Step 1 (Service deployment) and Step 3 (Finding working paths) correspond to the bootstrapping phase, where the initial service deployment and assignments of working paths are performed, aiming at minimizing the working path lengths. In Step 2 (Shared link protection), backup paths are found for each demand in a way to maximize the shared use of links similar to the shared protection phase. Note that the order of steps for finding working and backup paths are different than the optimization scheme, which is justified in more detail later on in the paper. Step 4 (Assigning remaining demands) plays a complementary role for Steps 1-3 to ensure that the working and backup paths are assigned for all demands with minimal path lengths and maximal backup capacity sharing. Lastly, Step 5 (Finding backup nodes) is to find alternative service deployment schemes utilizing the shortest available paths to set up in the case of failures, corresponding to the service migration phase. In the rest of this section, we describe each step in more detail.

Step 1 (Service Deployment): In this step, the services are assigned to physical nodes with sufficient resources. As each demand is defined between a pair of services, the locations of host nodes are restrained by the latency and data requirements of demands. The host nodes are selected starting from the ones with the highest connectivity, e.g., the highest number of direct neighbors, to ease finding disjoint working and backup paths afterward. For each pair of services utilized by a demand, we ensure that there is at least a path with sufficient link resources and latency cost for the respective demand. Among the alternative deployments, we select the closest nodes for a better quality of service in terms of latency and less link resource consumption at the end.

Note that a service can be utilized by multiple demands. In this case, only a single instance of the related service is placed to the selected node. Accordingly, this node should satisfy latency and data requirements of any demand utilizing that service.

Step 2 (Shared Link Protection Scheme): When some of the services are utilized by multiple demands as mentioned in the previous step, they form *chain of services* as it is also seen in Fig. 1. For instance, while a node hosting a service receives data for a traffic demand, that node can send data from the same service to another node to satisfy a different demand. When such services are allocated at physical nodes, it is convenient to define a communication backbone connecting and covering all those nodes to be shared by multiple demands in case of failures. In this step, we utilize our Secondary Backup Backbone (SBB) algorithm to define the chain of services whose host nodes can also be connected sequentially to form the backup backbone. That is, the service chain in the service overlay is also reflected as a chain of physical nodes forming a single backbone, i.e., a connected subgraph. In contrast to the order of optimization phases in Section III, instead of finding working paths first together with the service deployment, we apply a shared link protection scheme before assigning working paths. The reason is that the assignment of working paths restricts the available links to be used in backup paths significantly as they should be disjoint. However, when we maximize the shared use of backup links, i.e., by decreasing the number of used links and the total capacity, there are still sufficient resources left to establish shorter working paths.

For each service chain, SBB constructs a modified Steiner tree [44] on the physical network where each host node is considered as a terminal node and any intermediate node, which belongs to the tree, is a Steiner node. A Steiner tree is defined as a connected subgraph, e.g., a tree, including a set of given nodes, i.e., terminal nodes. All other nodes to be used to connect terminal nodes are called Steiner nodes. Generally, the construction of a Steiner tree refers to finding a minimal subgraph, or the shortest tree, having all terminal nodes with a minimum number of Steiner nodes, and it is known to be an NP-complete problem [45]. However, according to our initial experiments, finding the shortest tree eliminates the possible use of shortest paths as working paths afterward. Since the working paths are the most used ones until a failure occurs, keeping them shorter leads to a better QoS. Therefore, we use a simple heuristic to construct a *secondary* Steiner tree, which utilizes the shortest disjoint path to the shortest path between two nodes, i.e., the secondary shortest path, instead of directly using the shortest one. Algorithm 1 briefly describes those steps of SBB.

Iterating through the elements of $\mathrm{v}_{\mathrm{terminal}}$ including all host nodes obtained from the previous step, SBB forms the

Algorithm 1: Secondary Backup Backbone (SBB)

1 $\mathrm{v}_{\text{terminal}} \leftarrow [v_1, v_2, v_3 ... v_n]$
2 $\mathrm{v}_{\text{selected}} \leftarrow \varnothing$
3 $\mathrm{v}_{\text{Steiner}} \leftarrow \varnothing$
4 $v \leftarrow$ random node $\in V_{\text{terminal}}$
5 $\mathrm{v}_{\text{selected}} \leftarrow V_{\text{selected}} \cup \{v\}$
6 $\mathrm{v}_{\text{terminal}} \leftarrow V_{\text{terminal}} \setminus \{v\}$
7 **while** $\mathrm{v}_{terminal} \neq \varnothing$ **do**
8 $p^* \leftarrow \infty$
9 **for** $\mathrm{v}_1 \in V_{terminal}$ **do**
10 $\mathrm{v}_2 \leftarrow v_n \in V_{\text{selected}}$ closest to v_1
11 **if** v_2 is close enough to $\forall v_n \in V_{terminal}$ **then**
12 $p \leftarrow$ secondary shortest path between v_1-v_2
13 **if** p is shorter than $p*$ **then**
14 $p^* \leftarrow p$
15 $v \leftarrow v_2$
16 $\mathrm{v}_{\text{selected}} \leftarrow V_{\text{selected}} \cup \{v\}$
17 $\mathrm{v}_{\text{terminal}} \leftarrow V_{\text{terminal}} \setminus \{v\}$
18 $\mathrm{v}_{\text{Steiner}} \leftarrow V_{\text{Steiner}} \cup \{v_n \in p^*\}$

Algorithm 2: Mutually Disjoint Paths (MDP)

1 **for** $d \in D$ **do**
2 $\mathrm{v}_1, \mathrm{v}_2 \leftarrow$ Host nodes of d
3 $p_d^w \leftarrow$ Backup segment between $\mathrm{v}_1 - \mathrm{v}_2 \in G_{\text{Steiner}}$
4 **for** $p \in P_d$ **do**
5 **if** p *and* $\{p_q^w | \forall q \in H_d\}$ *disjoint* **then**
6 **if** p *and* p_d^b *disjoint* **then**
7 **if** p *satisfies* l_d **then**
8 $p_d^w \leftarrow p$

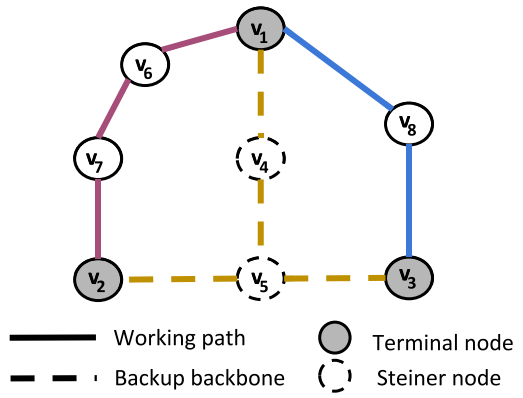

Fig. 5. Backup backbone (dashed black) with host (terminal) nodes $\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3$, Steiner nodes (dashed) $\mathrm{v}_4, \mathrm{v}_5$, and respective working paths (straight purple and blue). Two demands are defined between nodes v_1-v_2 and v_1-v_3. In this case, the links between v_1-v_4 and v_4-v_5 are shared between those demands.

secondary shortest paths between the nodes verifying that the latency requirement for each demand is satisfied for the backup communication through the backbone. In line 11, SBB ensures that v_2 does not violate such requirements for a demand between the services on v_2 and any other terminal nodes in $\mathrm{v}_{\text{selected}}$. Then, the nodes in the shortest path excluding the end hosts, e.g., terminal nodes, are added to the set $\mathrm{v}_{\text{Steiner}}$ as Steiner nodes. After connecting each node in $V_{\text{selected}} \cup V_{\text{Steiner}}$, SBB returns that secondary Steiner tree satisfying demand requirements.

Step 3 (Finding Working Paths): After forming the backup backbone in Step 2, we use the Mutually Disjoint Paths (MDP) algorithm to calculate working paths in this step. Algorithm 2 shows how MDP calculates the working paths for demands whose backup paths are defined in a particular backup backbone G_{Steiner}.

There are three essential aspects to be considered when calculating working paths. First, they should be mutually disjoint if the backup paths of the respective demands are shared. Even though we have formed a single backbone, each demand $d \in D$ uses only a segment of the backbone G_{Steiner}, i.e., a path between the nodes hosting the services of that demand, that could be shared or not. Therefore, when finding a working path, we first check if $d$utilizes any capacity shared with another demand $q \in H_d$ with an assigned working path w_q and ensure that they are disjoint (line 5). The second issue is, the working path of d, p_d^w should be disjoint to the respective segment of the backup backbone p_d^b that is used that demand (line 6). Lastly, p_d^w should satisfy the latency requirements of d (line 7). Note that similar to the optimization scheme, we have found all paths in advance and added them to the problem as an input. Therefore, checking the disjointness of paths is a matter of comparison between their nodes and links.

Fig. 5 shows an example construction of a backup backbone and the respective working paths. Initially, three services are deployed to v_1, v_2, and v_3 where two demands are defined between v_1-v_2 and v_1-v_3. Using additional Steiner nodes v_4 and v_5, SBB constructs the backup backbone among the nodes v_1-v_5. Accordingly, MDP calculates two disjoint working paths v_1-v_6-v_7-v_2 and v_1-v_8-v_3. Note that those paths are both mutually disjoint and concerning their relation with the backbone. In the backbone, the links between v_1-v_4 and v_4-v_5 are shared.

Step 4 (Assigning Remaining Demands): As a result of Step 2 and 3, the working and backup paths are calculated around backbones tightly for each service chain. In particular cases such as the lack of disjoint paths and limited available link resources, some demands may not be assigned according to the initial service deployment in Step 1. To cope with such scenarios, we utilize an improved version of our previous heuristic Random Deployment with Disjoint Paths (RDDP) in [10]. RDDP is a greedy heuristic consisting of two phases, service deployment and routing. In the first phase, it allocates the services to a randomly selected pair of nodes not hosting other services. If every node hosts at least one service, RDDP selects the node with the highest available resources. We improve RDDP to select the nodes whose secondary shortest path maximizes the use of shared capacity. In the second

phase, it allocates two node-disjoint paths for the inter-service demand between selected nodes, one for the main use and the other one as a redundant backup. If disjoint paths cannot be found between those nodes, they are reselected by following the same greedy approach.

Note that RDDP selects the feasible nodes and paths for each demand after a limited number of trials where the pairs of nodes are examined randomly as it is explained in [10]. The design parameter *limit* can be selected according to the network size as the number of possible 2-combinations of nodes is proportional. In our experiments, we applied *limit* $= 100$.

Step 5 (Finding Backup Nodes): Although we have established backup paths for protection against single link failures so far, any failure occurring in host nodes can still disrupt the communication as the hosted services would fail. In the last step of the heuristic, we utilize another heuristic from our former work, namely Backups with Secondary Redundant Path (BSRP). It simply finds an alternative node for each host node in the initial deployment to migrate its services in the case of a failure.

When an alternative node is selected to host a service, three criteria are important to satisfy the requirements of all demands which utilize that service: The alternative node should (i) have sufficient resource capacity to host the respective service, (ii) have required paths with sufficient capacity in-between the nodes hosting the other services for related demands, and (iii) be in a position to comply with the latency requirements of all related demands. In BSRP, we search for alternative nodes starting from the ones with the highest remaining resource capacity for each service. Among the candidates, the node with the minimum total length of paths to the other services for the respective demands utilizing the migrated service is selected. Eventually, the alternative paths to be used after a service migration consume network resources minimally.

V. Evaluation

To measure the performance of our optimization scheme and heuristic, we considered a number of scenarios and used a set of metrics. In this section, we present our experimental setup and discuss our numerical results in detail.

A. Experiment Setup

In this section, we describe (i) our computational resources and tools used to run our experiments, (ii) our topology and service overlay generation approach, (iii) what we measured and related parameters, and lastly (iv) the metrics we used for the comparisons.

1) Computational Resources: The optimization models were implemented in CPLEX 12.7.0, and all experiments were conducted in a server with 64-core Intel Xeon 2.10GHz CPU and 256GB RAM. The resource utilization varied for the different phases of the optimization scheme. For the largest instances of the problem, i.e., 35 nodes and 21 demands, phase 3 (service migration) kept the CPU utilization around the level of 80% for all the cores and used all available RAM. As we used pre-computed paths in the model, it also occupied an amount of memory proportional to the network size and connectivity. On the contrary, phase 1 and 2 were executed more easily using the CPLEX branch-and-bound method and utilizing the presolver.

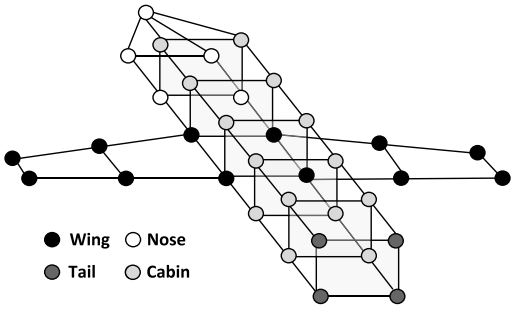

Fig. 6. Potential in-plane topology.

TABLE III
Characteristics of the Services That Are Used in the Experiments

Service Set	Resource demand	Criticality	Position	Quantity
S_1	Low	High	Nose, cabin, tail	$\|S\|/4$
S_2	Medium	Medium	Any	$\|S\|/4$
S_3	High	Low	Cabin, tail	$\|S\|/2$

2) Topology and Overlay Generation: We evaluated the performance of the optimization models and the heuristic for different types of topologies and service overlays. First, we created a potential in-plane topology shown in Fig. 6 as avionics is one of the key safety-critical domains to make use of service-based flexible network design. In this topology, we considered a cabin network that is interconnected with the nose, tail, and wings of the plane. Note that in traditional in-plane networks, there might be tens of end-systems, subsystems, and hundreds of signals between critical components [46]. Here, we included the cabin also giving services with higher traffic loads, e.g., infotainment, and combined it with the rest of the network to have a complete model of a connected aircraft. The entire topology has 35 nodes with different service-hosting capabilities and the average node degree of 3.7. Eventually, this network can be extended to any other mission-critical domain having varying traffic demands to be satisfied within a bounded latency.

For the given in-plane topology, we defined a service overlay with certain characteristics. Tables III and IV show those characteristics for the services and demands, respectively. In Table III, we present three types of services concerning their resource demand, criticality, possible position to be placed in the topology, and their quantity. The set S_1 consists of low resource demand and high criticality, e.g., control signals, that can be in nose, tail, and cabin. A quarter of all the services belongs to that type. S_2 represents medium resource demands and criticality services that can be placed anywhere. Lastly, S_3 consists of the services with high-resource demand and low-criticality to be placed in cabin and half of the services are defined in that type. We generated the exact values, e.g., traffic

TABLE IV
CHARACTERISTICS OF THE DEMANDS THAT ARE USED IN THE EXPERIMENTS

Demand Set	Traffic demand	Latency	Services	Quantity
D_1	Low	Time-sensitive	$S_1 \leftrightarrow S_1$	$\|D\|/4$
D_2	Medium	Best-effort	$S_1 \leftrightarrow S_2$, $S_2 \leftrightarrow S_2$, $S_3 \rightarrow S_2$	$\|D\|/4$
D_3	High	Best-effort	$S_3 \leftrightarrow S_3$	$\|D\|/2$

and latency requirements of demand, for each type of service randomly keeping their interrelation.

We considered three types of demands shown in Table IV utilizing the types of services from Table III. D_1 consists of demands with low data traffic and being time-sensitive, e.g., with tight latency constraints, and defined between the services of S_1, i.e., the most critical ones. D_2 represents the demands with medium data traffic with best-effort QoS between the services of S_1, S_2, and S_3. Lastly, D_3 represents the high-traffic demands, mostly defined for the cabin part. We defined half of the demands in D_3 as proportional to the number of nodes in the cabin part, and the rest of the demands were equally distributed between D_1 and D_2.

Apart from a very regular in-plane topology where the majority of the nodes have similar connectivity, we also used random networks with the same number of nodes and similar connectivity. As nodes do not have particular roles, e.g., either in nose, cabin, tail, or wings, we did not restrict the position of the services. That is, we used the same characteristics for the service overlay, excluding the positional constraints.

3) Measurements and Parameters: For most of the experiments, we measured *Optimal* and *Heuristic* values that represent the optimal and the heuristic's results for the given in-plane topology, *Optimal-R* and *Heuristic-R*, in contrast, show the results for random networks of the same size.

We evaluated the models and the heuristic for the increasing number of demands. In the end, we also show the scalability of the heuristic for the increasing number of nodes generating larger random topologies (50-70 nodes) with the average node degree of 2.8. In the scalability scenarios, we used a fixed-size service overlay with 50 demands. For each scenario, the experiments were repeated 30 times and the results are given with a 95% confidence interval. For all experiments, the optimality gap was defined as 5%, which means the results could deviate from the optimum at most by 5%.

4) Metrics: We used the metrics listed below to evaluate the performance of all phases of the optimization scheme and the heuristic.

Probability of Service Failure: It is the ratio of the number of services that cannot be migrated to an alternative node to all services.

Sharing Efficiency: It is the ratio of the difference between shared and dedicated backup capacity to the dedicated backup capacity without sharing. The latter is calculated by disabling capacity-sharing and reserving dedicated capacity on the configured backup paths. The sharing efficiency represents the capacity gain by sharing. We compare our capacity sharing approach with the dedicated one as it is a broadly-used approach for the redundancy in mission critical systems. Besides, this comparison helps to emphasize the improvements made by us following our preliminary work, where we used the dedicated backup capacity approach.

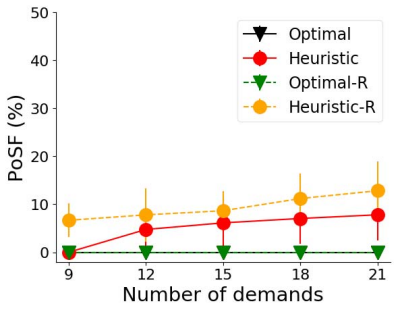

Fig. 7. Probability of service failure in case of a single node failure.

Shared Link Ratio: It is the ratio of the shared links to all backup links. The sharing ratio measures the efficiency of backup capacity allocation where the backup links can be shared only if the respective working paths are mutually disjoint. Any link utilized as a backup link by multiple demands is counted as a shared link.

Total Length of Working Paths: As Bootstrapping-ILP and Migration-LP optimize the total path length in terms of number of hops, this metric represents the total objective value of those two phases.

Backup Capacity Use: It is the total link capacity, i.e., bandwidth, required for all the backup paths. As Backup-ILP minimizes the use of backup capacity by utilizing shared capacity, this metric also represents the resulting value of the objective function.

Total Capacity Use: It is the total link capacity required for all the paths, i.e., both working and backup paths. Especially for the heuristic, it shows the efficiency of working path selection.

Prolongation Factor: Prolongation factor between two paths is the ratio of the length of one path to the other's length. We consider (i) backup to working path and (ii) selected backup to the ideal backup path prolongation factors to show the balance between backup and working paths, and the efficiency of the selected paths, respectively.

Solution Time: It is measured for each individual phase of the optimization scheme. As the heuristic solves the target problem in a neglectable time, e.g., seconds to a few minutes, the solution time is considered for only the optimization scheme to evaluate the increasing complexity by the number of demands.

B. Results

In this section, we present the experiment results using the selected metrics and scenarios for different topologies and service overlays.

1) Fault-Tolerance: Fig. 7 shows the probability of service failure (PoSF) during a single node failure as a function of the number of demands. As the optimal solution guarantees

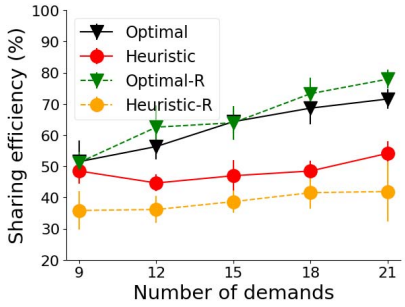

Fig. 8. Sharing efficiency.

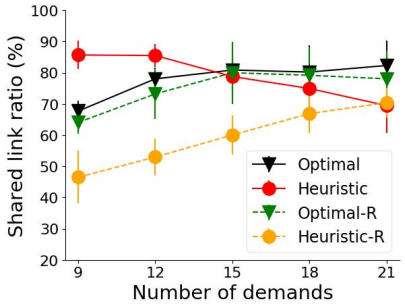

Fig. 9. Shared link ratio.

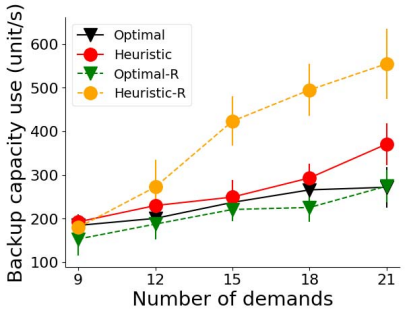

Fig. 10. Total backup capacity use.

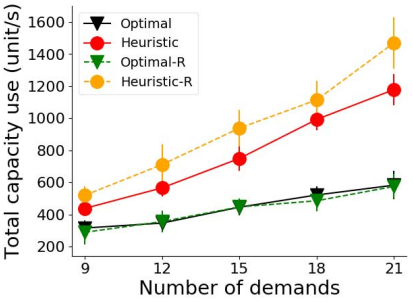

Fig. 11. Total capacity use.

to find a backup path for each demand against link failures and a backup node for each service against node failures, it protects the network against all single link failures and any single failure of the host nodes. Therefore, the optimal deployments at both in-plane and random topology result in 0% PoSF. Similarly, in our experiments, our heuristic is also successful to reserve the required capacity in backup paths for all the demands. However, in the last stage of the heuristic, there are some scenarios where it fails to find alternative nodes to migrate services due to (i) insufficient amount of node resource capacity, (ii) link capacity, or (iii) lack of suitable paths satisfying the latency requirements. For the in-plane topology and random networks, our heuristic keeps the PoSF below 5% and 10%, respectively. As the service deployment may spread through the network without node capability constraints and random connectivity, Step 5 in the heuristic fails more often to migrate all services in random networks satisfying especially case (iii) above.

Note that some mission-critical services cannot tolerate any failure at all and require replicated hardware or software to ensure seamless failover in case of single failures. Our heuristic can be considered as an additional fault-tolerance mechanism for such cases, thus avoiding replication costs. Furthermore, our optimization model can be used to compute the desired level of fault-tolerance for all services. Hence, it can be used during the network design stage to plan the whole backup scheme despite its longer solution time.

2) Sharing Efficiency: Fig. 8 shows the sharing efficiency, which is the gain of using the shared capacity instead of dedicated capacity, depending on an increasing number of demands. While the optimization scheme (only Backup-ILP) can utilize backup paths to decrease the required backup capacity by 50-75%, our heuristic gives steady results around 50% and 40% backup capacity savings for in-plane (*Heuristic*) and random (*Heuristic-R*) topologies.

Similar to sharing efficiency, Fig. 9 shows the shared link ratio to measure the effective use of backup paths for an increased shared capacity. The optimal deployment results in 60-80% of the backup links used by several backup paths at once. For the in-plane topology, our heuristic results in 70-85% shared link use, which is quite similar to the results of the optimization scheme. Note that since maximizing the shared link ratio is not the objective of the optimization scheme, the heuristic can give better results for small number of demands. However, with an increasing number of demands, while the result of the optimization scheme converges to 80% shared link ratio, our heuristic shows a decreasing trend as it gets harder to find Steiner trees promoting mutually disjoint working and backup paths. On the other hand, the establishment of disjoint working paths is easier even for an increasing number of demands in random networks as node capability constraints are neglected, and demands can be placed more flexibly. In that case, *Heuristic-R* results in 45-70% shared link ratio.

Considering Fig. 8 and Fig. 9, the number of shared links does not have to be proportional to the sharing efficiency as the latter also depends on the load and sharing links for small loads may not affect the efficiency so much.

3) Objective Functions: As presented in Section III, there are two different objectives for the three phases of our optimization scheme: (i) Minimizing the path lengths that are used as working paths at bootstrapping and after service migration, and (ii) minimizing the reserved resources for backup capacity. Figs. 10 and 12 show the comparison of the optimal

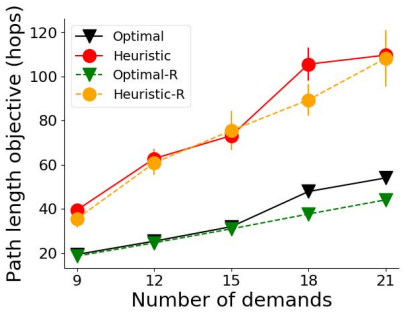

Fig. 12. Total length of bootstrapping and service migration phases.

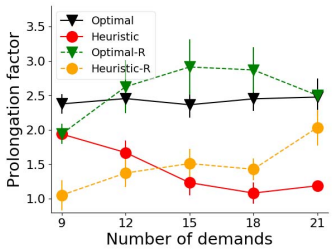

Fig. 13. Prolongation w.r.t working paths.

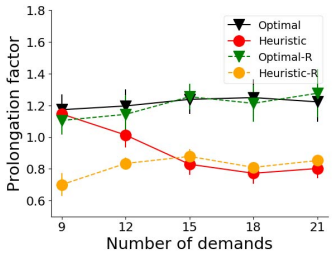

Fig. 14. Prolongation w.r.t. ideal backup paths.

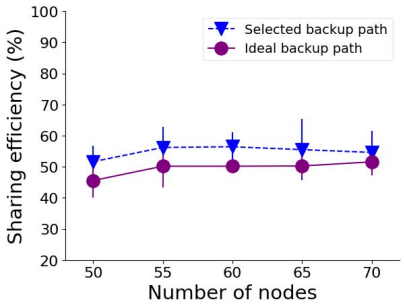

Fig. 15. Sharing efficiency with respect to ideal and selected backup paths.

solution and the results of our heuristic in terms of those two objectives.

Fig. 10 shows the total backup capacity utilization referring to the objective function (8). For the in-plane topology, our *Heuristic* give near-optimal results. However, for random topology, when we removed node capability restriction, e.g., any service can be placed to any node, the heuristic (*Heuristic-R*) may tend to spread the services to the farther nodes, which may result in longer Steiner trees. It is also related to the *limit* parameters as we examine a limited number of node combinations among the suitable nodes, e.g., having the capability to host a certain service, and select the best ones. This variety in possible changes can also be shown in the larger confidence interval in *Heuristic-R*. Such difference is also reflected in Fig. 11 and our heuristic requires a slightly higher capacity in random networks than in the more structured in-plane topology.

Fig. 12 shows the length of all working paths representing the results of the objective functions (1) and (15) for an increasing number of demands. For both in-plane and random topologies, the optimal results and the one from our heuristic scale proportionally. Even though we build secondary Steiner trees via SBB as described in Section IV instead of the shortest ones, some of the possible shortest paths are still used by the backup paths in the heuristic before constructing the working paths. Moreover, longer working paths might be utilized. However, as we embed latency requirements for demands to both working and backup finding processes, in our heuristic, the upper bound for the length of each working path is always restricted to satisfy such requirements.

4) Path Selection: The differences between the length of working and backup paths indicates the possible degradation in the QoS in case of a failure. That is, when a backup path is longer than the respective working path, shifting from a working to a backup path might result in an increased communication delay. Selecting shorter backup paths may consume resources that could be normally utilized by working paths. Fig. 13 shows the prolongation factor for backup and working paths (i.e., the ratio of the length of backup paths to working paths). The optimal solution on the in-plane topology (itOptimal), results in backup paths that are almost constantly 2.5 times longer than the corresponding working paths. In contrast, in random networks, the prolongation factor stays within 2 and 3 with larger confidence intervals. The structure of the random network seems to have a significant influence on the results. For the heuristic, as we first obtain backup paths by finding a backup backbone, some of the shorter paths are used for the backup before they can be chosen as working paths. It results in a greater number of working and backup paths of similar length, which manifests in prolongation factors in between one and two.

Fig. 14 shows the prolongation factor between the obtained backup paths and the *ideal* backup paths. An ideal backup path is defined as the secondary shortest path that is disjoint to the shortest path between two nodes. In the best case, where only a few demands exist in a large network, such a configuration for working and backup paths would give the best QoS and protect against any single link failure. As seen in Fig. 14, the optimal configuration provides 1.2-1.3 times longer backup paths in comparison to the ideal one. Our heuristic usually calculates shorter backup paths as it utilizes the backup backbone first. Therefore, the prolongation factor is 0.7-1.2, occasionally below 1.0.

5) Scalability: We evaluated the scalability of our heuristic for some of the selected metrics for an increasing topology size. Fig. 15 shows the sharing efficiency for 50 demands depending on the node count. Apart from calculating the total

TABLE V
PROBLEM SIZE AND SOLUTION TIME VALUES FOR THE GIVEN TOPOLOGY FROM FIG. 6 AND THE INCREASING NUMBER OF DEMANDS

# demands	Bootstrapping-ILP			Backup-ILP			Migration-LP		
	Variables	Constraints	Sol. Time	Variables	Constraints	Sol. Time	Variables	Constraints	Sol. Time
9	262314	344840	2.5m	223614	9593	41s	786744	1377508	18.4m
12	362184	515016	9.7m	298152	18474	54s	1676340	2995976	74.6m
15	483270	712880	16m	372690	19467	65s	3257940	6166825	3h
18	607572	938372	28.3m	447228	31989	103s	3663252	7277190	9h
21	741090	1191512	88.3m	521766	37935	204s	5736780	11920629	11.4h

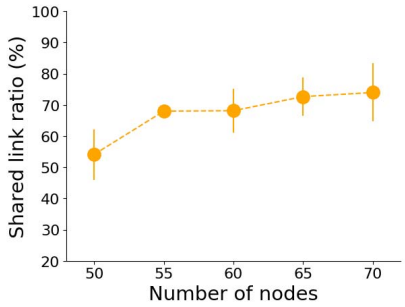

Fig. 16. Shared link ratio.

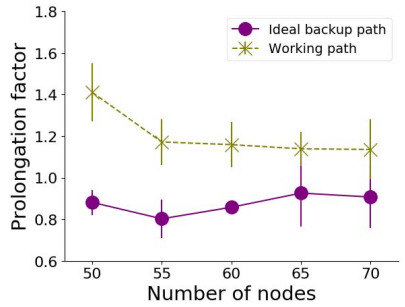

Fig. 17. Prolongation with respect to ideal backup paths and working paths.

backup load on the selected segments of the backup backbone considering shared and dedicated backup capacity (*Selected backup path*), according to our definition of sharing efficiency, we also considered another scenario where we reserved dedicated backup capacity on the ideal backup path (*Ideal backup path*) as it is defined in Section V-B4. The figure shows that our heuristic can again provide 50% capacity gain even for larger networks similar to the results for small topologies. Moreover, our shared backup capacity scheme seems to achieve 40-50% gain compared to a dedicated backup scheme for the ideal paths.

Fig. 16 shows that the shared link ratio increases up to 70% with increasing topology size implying larger set of potential backup paths. Lastly, we show the prolongation factor of shared backup paths with respect to the ideal backup paths and working paths combining in Fig. 17. As can be seen in the figure, for larger networks, our heuristic can find backup paths close to the ideal ones, i.e., with a prolongation factor of nearly 1.0, and better working paths with a lower prolongation factor 1.6 to 1.2 in comparison to smaller networks where this factor goes up to 2.0.

6) Solution Time: The phases of the optimization scheme require a considerable amount of time when increasing the size of the service overlay. As we have summarized in the complexity discussion of our previous work [10], finding the optimal configuration, which is resilient to all single node failures without considering any shared capacity, might take days as it is formulated as a single linear problem. Here, with three improvements, namely (i) dividing the whole optimization problem into three phases, (ii) finding disjoint backup paths against link failures instead of finding a new configuration for each failure and (iii) applying column generation method for the reduced problem (i.e., a fewer number of failure scenarios addressing only the failure of the host nodes), we improved the solution time considerably. Table V shows the problem size in terms of number variables and constraints and the solution time per each optimization phase.

Eventually, the service migration phase has more constraints and variables (when all the paths are added at the end of the column generation) than the other phases as it finds multiple configurations for different single node failure scenarios. Therefore, it was the decisive phase for the overall solution time with durations from 18 minutes to 11.5 hours. The overall solution took from 21 minutes to 14 hours, depending on the overlay size. It is a significant improvement for three times larger networks in comparison to our previous work [10].

VI. CONCLUSION

Mission-critical systems typically include several safety-critical services, which require considering their resilience against failures and attacks and already at the design stage. In this study, we present a service-based network design in which we embed an overlay of services and the traffic demands as well as QoS requirements between them in a mission-critical embedded network in a fault-tolerant manner. Extending our previous work in [10], we formulate the joint service deployment and routing problem as a series of optimization models to obtain a resource-efficient and resilient network model that provides fault-tolerance against single link and node failures leveraging shared backup protection and service migration schemes. To solve larger problem instances, we apply the column generation method in our optimization models. As finding the optimal solution for that joint problem is known to be NP-hard, we also propose a heuristic. We have evaluated the performance of the optimization scheme and the heuristic in different scenarios with various metrics. Our experiment results indicate that our heuristic can allocate near-optimal backup capacity offering up to 50% capacity gain compared to using dedicated backup capacity. Moreover, while our optimization scheme can find service configurations that

are completely fault-tolerant to single link and node failures, our heuristic can tolerate the single failure of 90% of all nodes.

In the future work, we plan to extend our objectives focusing on the different aspects of the network design, e.g., minimum energy consumption and minimum node or link deployment, as well as examining the impact of relevant parameters, e.g., the optimality gap. Moreover, although we assumed a single-failure model in this study as it is frequently considered in the literature, we intend to address more advanced models, e.g., correlated and cascading failures, under different assumptions such as shared risk link and node groups.

REFERENCES

[1] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, "Intelligent manufacturing in the context of industry 4.0: A review," *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.

[2] A. Koenig, "Integrated sensor electronics with self-X capabilities for advanced sensory systems as a baseline for industry 4.0," in *Proc. 19th ITG/GMA-Symp. Sens. Meas. Syst.*, 2018, pp. 1–4.

[3] O. Burkacky, J. Deichmann, G. Doll, and C. Knochenhauer, *Rethinking Car Software and Electronics Architecture*. New York, NY, USA: McKinsey & Co, 2020.

[4] *Automotive Virtual Platform Specification*, GENIVI, San Ramon, CA, USA, Jul. 2020.

[5] *Future Airborne Capability Environment (FACE) Technical Standard Edition 3.1*. San Francisco, CA, USA: Open Group, Jul. 2020.

[6] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: A survey," *IEEE Comm. Surveys Tuts.*, vol. 18, no. 1, pp. 553–576, 1st Quart., 2016.

[7] M. Nkomo, G. P. Hancke, A. M. Abu-Mahfouz, S. Sinha, and A. J. Onumanyi, "Overlay virtualized wireless sensor networks for application in Industrial Internet of Things: A review," *Sensors*, vol. 18, no. 10, p. 3215, 2018.

[8] K. Ogawa *et al.*, "Performance evaluations of IoT device virtualization for efficient resource utilization," in *Proc. Global IoT Summit (GIoTS)*, 2019, pp. 1–6.

[9] P. Karhula, J. Janak, and H. Schulzrinne, "Checkpointing and migration of IoT edge functions," in *Proc. 2nd Int. Workshop Edge Syst. Anal. Netw.*, 2019, pp. 60–65.

[10] D. Ergenc, J. Rak, and M. Fischer, "Service-based resilience for embedded IoT networks," in *Proc. 50th IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2020, pp. 540–551.

[11] G. Heiser, "Virtualizing embedded systems—Why bother?" in *Proc. 48th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2011, pp. 901–905.

[12] D. Espling, L. Larsson, W. Li, J. Tordsson, and E. Elmroth, "Modeling and placement of cloud services with internal structure," *IEEE Trans. Cloud Comput.*, vol. 4, no. 4, pp. 429–439, Oct.–Dec. 2016.

[13] D. Breitgand, A. Marashini, and J. Tordsson, "Policy-driven service placement optimization in federated clouds," Dept. Comput. Sci., IBM Res. Division, Armonk, NY, USA, Rep. H-0299 (H1102-014), 2011.

[14] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, "Online resource allocation, content placement and request routing for cost-efficient edge-caching in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1751–1767, Aug. 2018.

[15] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *J. Cloud Comput.*, vol. 7, no. 1, p. 4, 2018.

[16] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Proc. IEEE Int. Conf. Cloud Netw. (CloudNet)*, 2015, pp. 171–177.

[17] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "On orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[18] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 543–553, Sep. 2017.

[19] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, 2015, pp. 98–106.

[20] G. Lee, M. Kim, S. Choo, S. Pack, and Y. Kim, "Optimal flow distribution in service function chaining," in *Proc. ACM Int. Conf. Future Internet*, Jun. 2015, pp. 17–20.

[21] M. Médard, S. G. Finn, R. A. Barry, and R. G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Trans. Netw.*, vol. 7, no. 5, pp. 641–652, Oct. 1999.

[22] P. P. C. Lee, V. Misra, and D. Rubenstein, "Distributed algorithms for secure multipath routing in attack-resistant networks," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1490–1501, Dec. 2007.

[23] I. B. Barla, D. A. Schupke, M. Hoffmann, and G. Carle, "Optimal design of virtual networks for resilient cloud services," in *Proc. Int. Conf. Design Rel. Commun. Netw. (DRCN)*, 2013, pp. 218–225.

[24] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue, "Survivable virtual infrastructure mapping in virtualized data centers," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 196–203.

[25] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *Proc. IEEE Conf. Netw. Funct. Virtualization Softw. Defined Netw. (NFV-SDN)*, 2016, pp. 128–133.

[26] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Fault-resilient topology planning and traffic configuration for IEEE 802.1Qbv TSN networks," in *Proc. IEEE 24th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, 2018, pp. 151–156.

[27] X. Xie *et al.*, "Design for shared backup path protection based on content connectivity against disaster in elastic optical datacenter networks," in *Proc. 15th Int. Conf. Opt. Commun. Netw. (ICOCN)*, 2016, pp. 1–3.

[28] J. Rak, "Fast service recovery under shared protection in WDM networks," *J. Lightw. Technol.*, vol. 30, no. 1, pp. 84–95, Jan. 1, 2012.

[29] B. Todd and J. Doucette, "Demand-wise shared protection network design and topology allocation with dual-failure restorability," in *Proc. 11th Int. Conf. Design Rel. Commun. Netw. (DRCN)*, 2015, pp. 73–80.

[30] F. He, T. Sato, and E. Oki, "Survivable virtual network embedding model with shared protection over elastic optical network," in *Proc. IEEE Int. Conf. Cloud Netw. (CloudNet)*, 2019, pp. 1–3.

[31] J. C. Kumaran and M. Aramudhan, "A survey on resource allocation strategies in cloud," *Int. J. Reason. Based Intell. Syst.*, vol. 10, nos. 3–4, pp. 328–336, 2018.

[32] N. K. Pandey, S. Chaudhary, and N. K. Joshi, "Resource allocation strategies used in cloud computing: A critical analysis," in *Proc. IEEE Conf. Commun. Control Intell. Syst. (CCIS)*, 2017, pp. 213–216.

[33] X. Li and C. Qian, "A survey of network function placement," in *Proc. CCNC*, 2016, pp. 948–953.

[34] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Comput. Netw.*, vol. 133, pp. 212–262, Mar. 2018.

[35] C. Ou, J. Zhang, H. Zang, L. H. Sahasrabuddhe, and B. Mukherjee, "New and improved approaches for shared-path protection in WDM mesh networks," *J. Lightw. Technol.*, vol. 22, no. 5, pp. 1223–1232, May 2004.

[36] G. Ellinas, D. Papadimitriou, J. Rak, D. Staessens, J. P. G. Sterbenz, and K. Walkowiak, "Practical issues for the implementation of survivability and recovery techniques in optical networks," *Opt. Switch. Netw.*, vol. 14, pp. 179–193, Aug. 2014.

[37] G. P. Mccormick, "Computability of global solutions to factorable non-convex programs: Part I—Convex underestimating problems," *Math. Program.*, vol. 10, no. 1, pp. 147–175, Dec. 1976.

[38] S. Ramamurthy, L. Sahasrabuddhe, and B. Mukherjee, "Survivable WDM mesh networks," *J. Lightw. Technol.*, vol. 21, no. 4, pp. 870–883, 2003.

[39] J. Rak and D. Hutchison, *Guide to Disaster-Resilient Communication and Networks* (Computer Communications, Networks). Cham, Switzerland: Springer, 2020.

[40] J. Tapolcai, P. Ho, D. Verchere, T. Cinkler, and A. Haque, "A new shared segment protection method for survivable networks with guaranteed recovery time," *IEEE Trans. Rel.*, vol. 57, no. 2, pp. 272–282, Jun. 2008.

[41] M. Ruiz, M. Pióro, M. Żotkiewicz, M. Klinkowski, and L. Velasco, "Column generation algorithm for RSA problems in flexgrid optical networks," *Photon. Netw. Commun.*, vol. 26, nos. 2–3, pp. 53–64, 2013.

[42] C. Rocha and B. Jaumard, "A unified framework for shared protection schemes in optical mesh networks," *Pesquisa Operacional*, vol. 29, no. 3, pp. 533–546, 2009.

[43] L. S. Lasdon, "Duality and decomposition in mathematical programming," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 86–100, Jul. 1968.

[44] K. Makki and N. Pissinou, "The Steiner tree problem with minimum number of vertices in graphs," in *Proc. 2nd Great Lakes Symp. VLSI*, 1992, pp. 204–206.

[45] A. Biniaz, A. Maheshwari, and M. Smid, "On the hardness of full Steiner tree problems," *J. Discr. Algorithms*, vol. 34, pp. 118–127, Sep. 2015.

[46] B. Annighoefer, C. Reif, and F. Thieleck, "Network topology optimization for distributed integrated modular avionics," in *Proc. IEEE/AIAA 33rd Digit. Avionics Syst. Conf. (DASC)*, 2014, pp. 1–12.