Flexible Ethernet Traffic Restoration in Multi-layer Multi-domain Networks

Dahina Koulougli, Kim Khoa Nguyen, Mohamed Cheriet
Synchromedia - École de Technologie Supérieure, University of Quebec, Canada
Email: {dahina.koulougli.1}@ens.etsmtl.ca, {kim-khoa.nguyen, mohamed.cheriet}@etsmtl.ca

ICC 2021 - IEEE International Conference on Communications | 978-1-7281-7122-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICC42927.2021.9500270

***Abstract*—Recently, Flexible Ethernet (FlexE) has emerged as a new transmission technology allowing the flexible utilization of optical transport. This flexibility helps improve network ability against failures. FlexE recovers from a physical link (PHY) failure by migrating traffic to a new PHY. However, this task is costly, especially for critical failures or when the network is under high utilization. In this paper, we investigate the FlexE Traffic Restoration (FTR) problem that aims to maintain high network utilization by the fast recovery of FlexE clients with the minimum cost using the spare capacity in the already deployed PHYs. High network utilization can be obtained by rerouting of FlexE subgroups, moving clients to another subgroup, and shifting the clients' slots in the same subgroup without traffic disruption. We formulate the FTR optimization problem and solve it in polynomial time using learning theory and approximation. Experiments carried out in a real testbed show the proposed solution recovers 63% more traffic than baseline restoration schemes.**

***Index Terms*—MLMD networks, Flexible Ethernet, reliability, traffic restoration, learning theory.**

I. Introduction

Multi-layer Multi-domain (MLMD) is an evolved architecture adopted in the core 5G and beyond networks. MLMD is the integration of IP over the Optical Transport Network (OTN) and Dense Wavelength Division Multiplexing (DWDM) layers/domains [1]. The orchestration of the MLMD network through Software Defined Networking (SDN) and the Hierarchical Path Computation (H-PCE) allows the optimization of end-to-end routing [2]. A key feature of a MLMD orchestration platform is traffic restoration that ensures network resilience and reliability against failures. Unlike protection where the connections are recovered over pre-reserved paths, restoration dynamically computes the backup paths when the failure occurs without prior reservation of resources thus reducing the resource waste. [3].

Multi-layer restoration has been investigated in prior research, in which a centralized PCE computes backup paths sequentially or in a bulk to recover the requests [4]. Since a single failure could interrupt multiple established paths, the bulk path computation is more efficient in serving the concurrent requests because it optimizes the overall restoration bandwidth instead of sequential recovery that suffers from resource contention and inefficient grooming. However, the centralized PCE faces scalability and security issues that prevent its deployment in multi-domain networks.

Restoration in multi-domain networks is performed either by a distributed Backward Recursive (BRPCE) or a Hierarchical (H-PCE), and classified either as link or path-based [5]. In link-based restoration, the affected traffic is rerouted from the failed link to other links, whereas, in path-based scheme, traffic is redirected toward a new disjoint path. H-PCE is more efficient as the optimal sequence of domains is dynamically selected instead of predefined by the service provider.

In reality, the rerouting should be done in a Make Before Break (MBB) manner to avoid traffic disruption, where traffic cannot be recovered until setting up the backup paths has finished. Therefore, we should proceed with consistent network updates that carefully schedule the updates from the failed to the backup state given the bulk path computations [6]. In [7], such updates are done gradually instead of one-shot or parallel updates to avoid transient congestion at the individual switches which consume an operation delay to apply the updates. Unlike the aforementioned work, this paper addresses the inter-domain restoration to overcome single or multiple FlexE PHY failures in MLMD networks subject to new restoration metrics and constraints.

FlexE connects IP to the optical layers through the aggregation of multiple links to optimize the inter-domain efficiency [8]. FlexE carries clients over PHYs that are aggregated to form a FlexE group in a multi-layer network. In MLMD networks, the FlexE group is partitioned into multiple FlexE subgroups, each traversing a different domain (IP, OTN, or DWDM) through an intra-domain path. Prior research leverages on FlexE for traffic routing in multi-layer IP over fixed-DWDM and Elastic Optical Networks (EON) [9], [10]. In our previous work [2], we integrated FlexE in an MLMD network and showed significant improvements in terms of throughput and latency. In [2], a Routing and FlexE Assignment (RFA) mechanism has been proposed to find an optimal route for each FlexE subgroup and an assignment for each FlexE client. However, the reliability has not been considered so far. In this paper, we will leverage on low latency and high link aggregation capabilities of FlexE to address the rerouting issue in traffic restoration.

Fig. 1 illustrates an example of FlexE traffic restoration in an MLMD network, where there are two FlexE subgroups: (i) SG-1 has one spare PHY of 100 Gb/E that is not yet provisioned (PHY4-1) and 3x100GbE provisioned PHYs (PHY1-1, PHY2-1 and PHY3-1) carrying FlexE clients A of 150 Gb/s (in red color), B of 50 Gb/s (blue), C of 10 Gb/s (green) and D of 40 Gb/s (yellow); and (ii) SG-2 has one 1x100 GbE PHY carrying FlexE client E of 25 GB/s (purple). Upon the failure of PHY2-1 shown in Fig. 1(b), the affected slots can

978-1-7281-7122-7/21/$31.00 ©2021 Crown

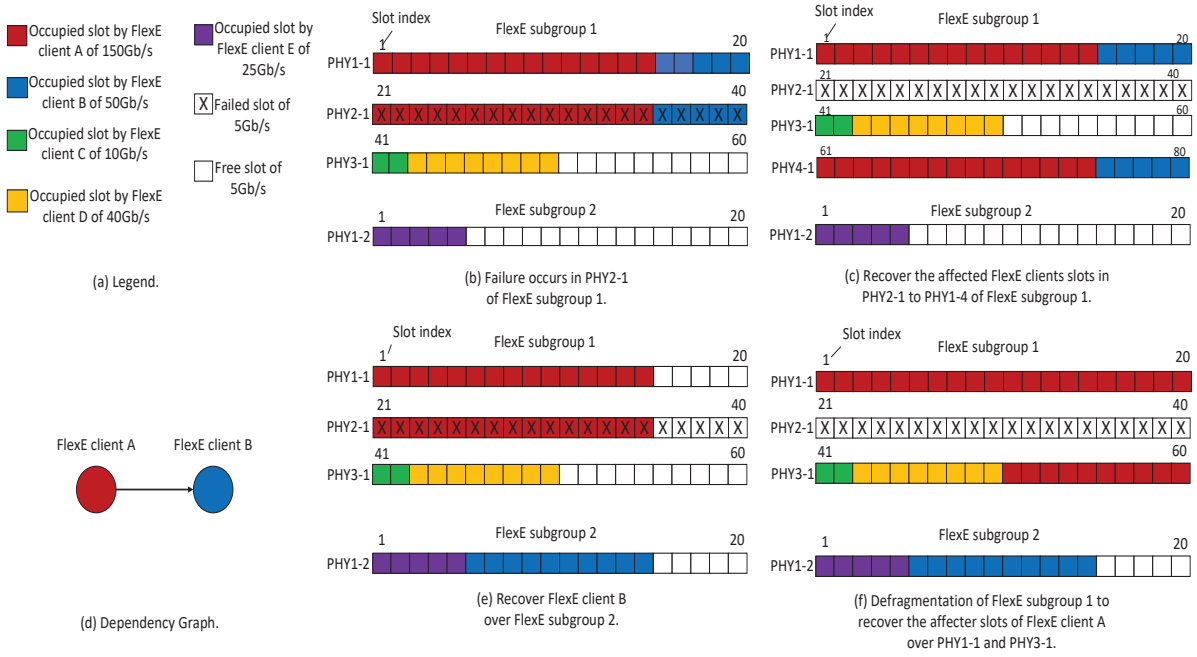


Fig. 1: Example of FlexE clients restoration over FlexE subgroups in an MLMD network

be recovered through link-based restoration over PHY4-1 in Fig. 1(c). If a path-based recovery is used, FlexE traffic of SG-1 is rerouted over a new FlexE subgroup (e.g., provisioning SG-3). However, both mechanisms are costly in terms of delay and resource provisioning. A more efficient recovery may exploit the spare capacity of the already deployed PHYs to maintain high network utilization without changing the intra-domain connections. Such restoration scheme achieves a congestion-free plan by a 2-step recovery: (i) recover FlexE client B by moving it to FlexE subgroup SG-2 in Fig. 1(e); and (ii) recover FlexE client A over its initial subgroup SG-1 in Fig. 1(f).

Moving client B to subgroup SG-2 creates a fragment in PHY1-1, which is common in finer granularity slot-blocks assignment problems such as fragmentation of file systems in computer storage or spectrum fragmentation in EON [11]. The main reason for optical spectrum fragmentation is the slot contiguity constraint. Fortunately, FlexE clients can be recovered from non contiguous FlexE slots that satisfy the ascending order of slots defined as follows.

Definition 1: Given a FlexE client, if its k^{th} slot denoted by s_k is affected by the failure. s_k must be recovered in the FlexE subgroup over a slot s_j with index j higher than the index k' of the predecessor client slot $s_{k'}$ and lower than the index k'' of successor assigned client slot $s_{k''}$, thus $k' < j < k$". We refer this as FlexE slot ordering.

Defragmentation techniques aim to minimize the service disruptions either by proactive or reactive rearrangement of the spectrum [11]. Proactive defragmentation consolidates the spectrum to minimize fragmentation however, frequent reconfigurations are redundant and costly. Instead, reactive defragmentation rearranges the spectrum when necessary to afford the rejected requests. As shown in Fig. 1(f), reactive defragmentation is performed after the occurrence of the failure to recover FlexE client A over PHY1-1 and PHY3-1.

In this paper, we address the FlexE Traffic Restoration (FTR) problem in the parent-PCE facing the issues of limited intra-domain information and FlexE fragmentation regarding the slot ordering constraint. The main contributions of this paper are as follows.

- We formulate the FlexE Traffic Restoration (FTR) problem as a Mixed Integer Nonlinear Program (MINLP).
- We derive an efficient algorithm based on learning theory and approximation to practically solve the FTR problem.
- We evaluate the performance of the solution over RFA and Disjoint Shortest Path (DSP) in a real testbed.

II. FlexE Traffic Restoration (FTR) problem

A. System Model

We model the MLMD network as a graph $G = (V, E)$, where $n \in V$ is a network node and $l \in E$ is a network link. We assume three FlexE subgroups that are configured according to each layer IP, OTN and DWDM. Each FlexE subgroup g is already assigned one FlexE tunnel p in the initial assignment state $y_{g,p}^{t=0}$ and carrying $\sum_{i,j,k} x_{ij,g^k}^{t=0}$ FlexE clients slots over $z_{g,p}^{t=0}$ PHYs. Each FlexE tunnel traverses one layer over an intra-domain IP path or a deployed lightpath in lower layers. The notation $t = 0$ denotes the initial state before the failure occurs. The recovery of clients must be done in a set of steps to achieve a congestion-free plan. Table I summarizes the notations used in this paper.

B. Problem formulation

FlexE Traffic Restoration (FTR) problem aims to find the optimal rerouting scheme satisfying the MBB policy. The rerouting in FTR refers to finding a new assignment for each FlexE client in the same or a different subgroup and eventually a new path for each FlexE subgroup. The MINLP formulation is given in the following.

$$\text{Max} \quad \Psi_u - \epsilon.\ \Psi_w + \sum_l c_{l,spare} \tag{1}$$

$$\text{S.t.} \quad \frac{d'_i}{c_{slot}} \leq \sum_{j,g,k} x^t_{ij,g^k} \leq \frac{d_i}{c_{slot}} \qquad \forall i,\ \forall t \tag{2}$$

$$\sum_{i,j,k} x^t_{ij,g^k}.\ c_{slot} \leq \sum_p z^t_{g,p}.\ c_{phy} \qquad \forall g,\ \forall t \tag{3}$$

$$\sum_{(g,k)\vee(i,j)} x^t_{ij,g^k} \leq 1 \qquad (\forall i,\ \forall j) \vee (\forall g,\ \forall k) \tag{4}$$

$$x^t_{ij,g^k} + x^t_{ij',g'^{k'}} \leq 1 \qquad \forall i,\ \forall j,\ \forall j',\ \forall k,\ \forall k',\ \forall t,\ \forall (g \neq g') \tag{5}$$

$$c_{l,spare} + \sum_{g,p} max\{z^t_{g,p}, z^{t+1}_{g,p}\}.\ I_{p,l}.\ c_{phy} \leq r_l \qquad \forall l,\ \forall t \tag{6}$$

$$\sum_p y^t_{g,p} \leq 1 \qquad \forall g,\ \forall t \tag{7}$$

$$z^t_{g,p} \leq y^t_{g,p}.\ rphy_g \qquad \forall g,\ \forall p,\ \forall t \tag{8}$$

$$x^t_{ij,g^k} + x^t_{ij',g^{k'}} \leq 1 \qquad \forall i,\ \forall g,\ \forall t,\ \forall (k > k'),\ \forall (j < j') \tag{9}$$

1) Objective: The objective of FTR (Eq. 1) is to maximize the restoration throughput $\Psi_u = \sum_{i,j,g,k,t} x^t_{ij,g^k}.\ c_{slot}$, minimize the latency Ψ_w and the provisioning cost of PHYs achieved by maximizing the spare capacity on links. Ψ_w is the sum of the update δ_{upd} and FlexE reconfiguration δ_{def} delays.

The update delay is the overall operation delay for updating FlexE tunnels on the nodes. The node's operation delay is defined in [7] and is equal to 0 when a FlexE subgroup does not change its initial tunnel, whereas, if the subgroup is rerouted over a new tunnel then the operation delay equals to a modification delay t_{md} in the ingress tunnel node and an insertion delay t_{in} in each of the internal tunnel nodes. However, the modeling in [7] is for one-shot updates and thus no valid in a congestion-free plan. We model the update delay with XOR function that captures whether a subgroup is rerouted between each two-succeeding steps as follows.

$$\delta_{upd} = \frac{1}{2}.\ \sum_{g,p,t} \left[y^t_{g,p}(1 - y^{t+1}_{g,p}) + y^{t+1}_{g,p}(1 - y^t_{g,p})\right] \sum_v t_{v,p}.\ \omega_v \tag{10}$$

FlexE reconfiguration delay σ_{def} is the total delay to reconfigure (move to another subgroup or defragment in the same subgroup) FlexE clients in a set of steps. Since FlexE clients are reconfigured in parallel in each step t, the total reconfiguration delay is given as follows.

$$\sigma_{def} = t_r.\ \sum_t \max_i \left\{ \sum_g sgn(\lambda^t_{i,g}) \right\} \tag{11}$$

Where t_r is the time to reconfigure FlexE clients, sgn is the sign function and $\lambda^t_{i,g}$ detects the clients slots that are reconfigured between each two-consecutive steps.

$$\lambda^t_{i,g} = \frac{1}{2}.\ \sum_{j,k} \left[x^t_{ij,g^k}(1 - x^{t+1}_{ij,g^k}) + x^{t+1}_{ij,g^k}(1 - x^t_{ij,g^k})\right] \tag{12}$$

2) Constraints: Eq. (2) ensures at each reconfiguration step t, the number of slots assigned to each FlexE client is at least the number of allocated slots not affected by the failure and should not exceed the number of slots assigned before the failure. Eq. (3) states that the total clients carried over a FlexE subgroup should not exceed its capacity. Eq. (4) is the slot assignment restriction constraint that assigns each FlexE client slot to one FlexE subgroup slot, likewise, each FlexE subgroup slot can be assigned to no more than one FlexE client slot. Eq. (5) is the assignment restriction to one FlexE subgroup for each FlexE client. Eq. 6 is the capacity constraint on the links which limits the worst link load denoted by $\sum_{g,p} max\{z^t_{g,p}, z^{t+1}_{g,p}\}.\ I_{p,l}.\ c_{phy}$. Such case happens when some FlexE subgroups do not free up some of their PHYs yet but should do so while FlexE subgroups that require new PHYs have been reconfigured [12]. Eq. (7) is the co-routing constraint, where FlexE subgroup traffic can be rerouted over one tunnel. Eq. (8) ensures that the total number of PHYs allocated to a FlexE subgroup cannot exceed the number of its available PHYs. Eq. (9) denotes the slot ordering constraint, where slots for each FlexE client must be recovered in ascending order to the free slots with ascending order positions in the FlexE subgroup.

TABLE I: Notations

Notation	Description
d_i	Flow demand for FlexE client i before the failure
d'_i	Flow demand for FlexE client i after the failure
r_l	Available capacity of link l after PHY failure
c_{slot}	Calendar slot granularity
c_{phy}	Capacity of a PHY
$rphy_g$	Residual number of candidate PHYs for FlexE subgroup g
$I_{p,l}$	Equals to 1 if link l in FlexE tunnel p
ω_v	weight of node v based on its traffic load
$t_{v,p}$	Equals to modification delay t_{md} if v is ingress node of p; or to insertion delay t_{in} if v is an internal node of p; 0 otherwise
Decision variables	
x^t_{ij,g^k}	Binary variable equals to 1 if the jth slot of FlexE client i is assigned to the kth slot of FlexE subgroup g at step t
$y^t_{g,p}$	Binary variable equals to 1 if FlexE tunnel p is assigned to FlexE subgroup g at step t
$z^t_{g,p}$	Integer variable equals to the number of PHYs assigned to FlexE subgroup g over FlexE tunnel p at step t
$c_{l,spare}$	Linear variable denotes the spare (margin) capacity on link l

III. FlexE Traffic Restoration (FTR) Algorithm

The proposed MINLP for the FTR problem is known to be NP-hard. To achieve timely recovery, we design an algorithm

consisting in two phases: Firstly, off-line learning the intra-domain delay (δ_{upd}) and the maximum number of PHYs that can be newly provisioned during failures (III-A), then a heuristic for on-line traffic restoration (III-B).

A. Off-line learning phase

The FTR problem involves the reconfiguration of clients over FlexE subgroups and the rerouting of paths. The rerouting happens when FlexE clients are moved to another FlexE subgroup with sufficient spare PHYs for recovery but the current intra-domain path carrying this subgroup is unable to recover the affected clients. The main issue is that the parent-PCE has no knowledge over the update delay nor the maximum number of PHYs that can be used for restoration due to its limited view over the intra-domain paths. Although the parent-PCE can approximate the optimal routing by having an abstract topological view, it is computational extensive and inefficient for rerouting [2]. Therefore, the parent-PCE learns the maximum number of PHYs that can be used for restoration as well as the intra-domain delay.

Algorithm 1 computes the maximum number of spare PHYs and the weight of FlexE subgroups. We use a stochastic greedy algorithm that starts by setting the number of PHYs $\hat{z}$ to a zero vector and then gradually improves it by iteratively adding of a PHY for each FlexE subgroup until the maximum number of spare PHYs $n_{g,spare}$ is reached [13]. At each iteration, the same number of PHYs z^t is used for training over multiple time slots to estimate the overall performance value $\hat{\Psi}$. The parent-PCE estimates the value of $\hat{\Psi}$ by averaging all the performance values $\Psi_{t,u,w}(z^t)$ observed over τ time steps from the child-PCEs. Then, $\hat{\Psi}$ is computed by summing up the observed throughput $\Psi_{t,u}(z^t)$ and weight $\Psi_{t,w}(z^t)$ which refers to the overall intra-domain delay. The algorithm returns the estimated number of PHY $\tilde{z}$ with the highest performance gain and updates the weight $\tilde{w}$ of subgroups.

The running time of Algorithm 1 is $T = \tau.\ \sum_g (1 + n_{g,spare})$ with $n_{g,spare} \leq n_{phy}$, thus the time complexity is $\mathcal{O}(\tau.\ |G|)$.

B. On-line phase

Algorithm 2 implements the following steps to quickly recover FlexE clients over FlexE subgroups instantly using the parameters learned off-line (maximum number of PHYs and FlexE subgroups weights) by Algorithm 1.

1) Computing the new network state: Algorithm 2 computes a new network state x' initialized at first to all-zero when a failure occurs (line 2). The residual capacity in the already deployed PHYs is computed for each FlexE subgroup g after deallocating all the affected clients (line 3). The spare capacity for restoration is then computed for each FlexE subgroup g, $\tilde{z}_g$ is the maximum number of PHYs learned off-line, z_g is the current number of PHYs carrying FlexE traffic (line 4). The

Algorithm 1: Off-line FTR learning

```
1  Initialize ẑ = z^0;  z̃ = 0;  w̃ = 0
2  for each FlexE subgroup g ∈ G do
3      Try out z^t = ẑ and observe Ψ_{t,u,w}(z^t)
         ∀t ∈ {1, .., τ}
4      Estimate Ψ̂(ẑ) = (1/τ) Σ_{t=1}^{τ} (Ψ_{t,u}(z^t) − ϵ. Ψ_{t,w}(z^t))
5      n_{g,spare} = n_{phy} − ẑ_g
6      for each PHY p picked from 1 to n_{g,spare} do
7          Set ẑ' = ẑ where ẑ'_g = ẑ_g + 1
8          Try out z^t = ẑ' and observe Ψ_{t,u,w}(z^t)
             ∀t ∈ {pτ + 1, ..., pτ + τ}
9          Estimate
             Ψ̂(ẑ') = (1/τ) Σ_{t=pτ+1}^{pτ+τ} Ψ_{t,u}(z^t) − ϵ. Ψ_{t,w}(z^t)
10         Set D(ẑ_g, ẑ'_g) = Ψ̂(ẑ') − Ψ̂(ẑ)
11     end
12     Update z̃ where z̃_g = argmax_{ẑ'} D(ẑ_g, ẑ'_g)
13     Update w̃ where w̃_g = Ψ_{t,w}(z̃)
14 end
15 Return z̃;  w̃
```

LP-relaxation is solved to find a fractional new assignment for each affected client (line 5) given in the following.

$$\text{Minimize} \quad \sum_g \tilde{w}_g.\ \rho_g \tag{13}$$

$$\text{Subject to} \quad \sum_i x_{i,g}.\ d_i \leq r_g + \rho_g \qquad \forall g \tag{14}$$

$$\sum_g x_{i,g} = 1 \qquad \forall i \tag{15}$$

$$0 \leq x_{i,g} \leq 1,\ 0 \leq \rho_g \leq \tilde{c}_{g,spare} \qquad \forall i, g \tag{16}$$

Eq. (13) aims to recover as much traffic over subgroups that experience the least delay and minimize the spare capacity ρ_g. Eq. (14) denotes the capacity constraint on FlexE subgroups, where the total traffic recovered over g does not exceed its spare capacity. Eq. (15) is the assignment restriction of each FlexE client to one FlexE subgroup. Finally, Eq. (16) denotes the domain constraints on linear variables $x_{i,g}$ and ρ_g.

The algorithm then applies a deterministic rounding to convert the fractional assignment obtained by the LP-relaxation to an integral one (lines 6-17). The residual capacity of each FlexE subgroup g is updated by ρ_g (line 6). We compute the pseudo-utility u_i of each client (line 7) and sort FlexE subgroups in the decreasing order of u_i (line 8). The algorithm iterates over FlexE clients and subgroups until a feasible assignment is reached (line 9-17).

2) Scheduling the network updates: The LP-relaxation provides one-shot updates, therefore, we need to carefully schedule the update from the initial state x to the target state x' (lines 18-20). Dionysus [6] achieves a consistent update ordering, where a path update cannot be started at the ingress node until the updates in the transit nodes are all done. Therefore, the parent-PCE should wait for the intra-domain updates from all child-PCEs to finish. Note that the intra-domain updates

Algorithm 2: On-line FTR

1 **Step 1: Compute the new network state**
2 Initialize $x' = 0$
3 $\forall g : r_g = z_g.\ c_{phy} - \sum_j x_{j,g}.\ d_j + \sum_i x_{i,g}.\ d_i$
4 $\forall g : \tilde{c}_{g,spare} = (\tilde{z}_g - z_g).\ c_{phy}$
5 $x' \leftarrow$ Solve LP-relaxation in Eq. (13)-Eq. (16)
6 $\forall g : r_g = r_g + \rho_g$
7 $\forall i : u_i = \sum_g x'_{i,g}.\ d_i$
8 $I \leftarrow$ sort FlexE clients in decreasing order of u_i
9 **for** *each client i in sorted list I* **do**
10 $G \leftarrow$ sort FlexE subgroups in decreasing order of $x'_{i,g}$ and increasing order of r_g
11 **for** *each FlexE subgroup g in sorted list G* **do**
12 **if** $d_j \leq r_g$ **then**
13 Set $x'_{i,g} = 1$ and $x'_{i,g'} = 0;\ \forall g' \neq g$
14 Update $r_g = r_g - d_i$; **Break**
15 **end**
16 **end**
17 **end**
18 **Step 2: Schedule the network updates**
19 **Child-PCEs:** schedule the intra-domain path updates for FlexE subgroups satisfying $\rho_g > 0$
20 **Parent-PCE:** waits until all the intra-domain updates are finished to schedule the inter-domain updates
21 **Step 3: FlexE defragmentation**
22 Perform reactive defragmentation to recover FlexE clients over their initial subgroups.

are applied only when some FlexE subgroups require setting up new PHYs ($\rho_g > 0$) that the current path cannot fulfill (line 19). In such a case, the Dionysus mechanism is performed at IP, whereas, an extension of this mechanism [14] is performed at the optical layers. Note that when $\delta_g = 0$, the updates are only performed at the inter-domain without affecting the intra-domain paths. To schedule the inter-domain updates (line 20), the parent-PCE constructs a dependency graph to capture the dependencies between the affected clients [15]. Fig. 1(d) illustrates the dependency graph, where a node refers to an affected FlexE client and a directed link between the nodes refers to the dependency between the corresponding clients. As shown in Fig. 1(d), there is a directed link between FlexE client A and FlexE client B which means client A depends on client B. Therefore, client A cannot be recovered over PHY1-1 in FlexE subgroup 1 until moving client B to FlexE subgroup 2 has finished to avoid traffic disruption.

3) FlexE defragmentation: To restore the clients that stand in their initial FlexE subgroups after computing the new state we have to defragment FlexE subgroups to ensure the slot ordering constraint in Eq. (8). We adopt the reactive hitless defragmentation strategy, in which clients in the failed subgroups are shifted in two phases following two-opposite directions known as Δ-defrag and ∇-defrag as in [11]. In Δ-defrag, clients slots in the first failed PHY are shifted to the lowest possible free slots of the subgroup to consolidate the PHY usage. In Fig. 1(e), the affected slots of FlexE client A are shifted from slot indexes 21-25 in PHY2-1 to lower free slot indexes 16-20 in PHY1-1. In ∇-defrag, clients slots starting from the uncovered ones in the failed PHY until the last involved slot in the subgroup are all moved toward higher free slots to recover as much as clients. In Fig. 1(e), the affected slots of FlexE client A are shifted from slot indexes 25-35 in PHY2-1 to higher free slot indexes 50-60 in PHY3-1.

The FTR algorithm experiences $\mathcal{O}(|G|.\ |I|^{3.5}.\ |L|^2)$ time complexity in Step 1, $\mathcal{O}(|V^d| + |E^d|)$ in Step 2, where $|V^d|$ and $|E^d|$ are the number of nodes and edges in the dependency graph, and $\mathcal{O}(|G|.\ |I|.\ log|I|)$ in Step 3. Therefore, the overall algorithm complexity is $\mathcal{O}(|G|.\ |I|^{3.5}.\ |L|^2)$, the relevant running time factor for solving the LP-relaxation in Eq. (13)-Eq. (16).

IV. Performance Evaluation

We have implemented the FTR in a real testbed and evaluate its performance over RFA [2] and H-PCE Disjoint Shortest Path (DSP) recovery [5] in terms of restorability, latency and provisioning cost of PHYs.

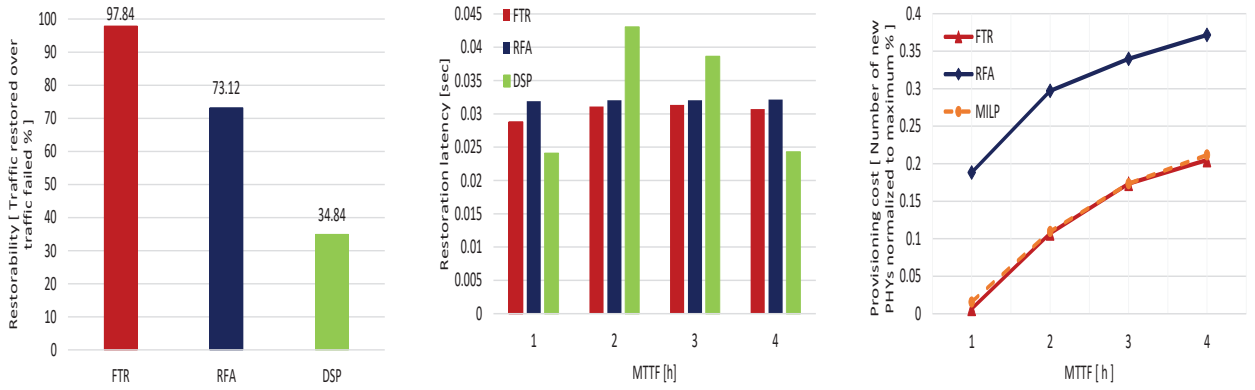
A. Prototype implementation

The MLMD testbed at TELUS-Ciena Lab in Edmonton, AB, has one ingress/egress pair of boundary nodes at each layer/domain as described in [2]. Layer 2 (IP domain) has four Juniper routers connected through 40 and 100 GbE links. Layer 1 (OTN domain) contains four Fujitsu optical switches linked through $OTU4$ (100G wave). Layer 0 (DWDM domain) is composed of three Ciena 6500 DWDM platforms connected by 400G wave capacity links. The boundary links between layers support 8x100 GbE FlexE PHYs per link. There is a child-PCE built-in each domain-controller and a parent-PCE (MLMD-PCE) built in the orchestrator.

FlexE clients arrive at the system following the Poison process and FlexE standard interval rates defined as $[\{10\}, \{40\}, \{n * 25\}]$ Gb/s, where $n \leq 4$ [8]. The Mean Time To failure $(MTTF)$ is set to 1 hour, the time at which the failure is simulated by randomly tearing down a set of inter-domain links (PHYs). The number of failure F generated at each hour is proportional to the number of hour (e.g. $MTTF = 1\ h$ leads to $F = 1$ and $MTTF = 2\ h$ results in $F = 2$). We set $t_{in} = 5\ ms$, $t_{md} = 10\ ms$ as in [7] and $t_r = 15\ ms$ [8]. Each point in the graphs generated in Fig. 2 results from 100 iterations.

B. Traffic restoration results

Fig. 2(a) shows the restorability of FTR, RFA and DSP schemes, measured as the ratio between the recovered traffic and the failed one. The results show that FTR recovers 97.84% of traffic, a performance which is 24.72% higher than RFA and 63% higher than DSP. The relevance of FTR against RFA stems from its ability to process the updates in a set of steps, whereas, RFA suffers from traffic loss during updates since the reconfiguration is done in one-shot an thus hits existing clients. Besides, FTR leverages on FlexE support and



(a) Restorability (b) Latency (c) Provisioning cost

Fig. 2: Comparative results of FTR, RFA and DSP in the MLMD testbed in terms of (a) restorability; (b) latency; and (c) provisioning cost of PHYs.

redistributes traffic more efficiently than DSP that selects the shortest disjoint path to recover FlexE clients.

Fig. 2(b) illustrates the restoration latency of FTR, RFA and DSP schemes over the number of failures and time. As shown, RFA experiences higher delay than FTR since the re-optimization in RFA involves moving huge traffic while FTR moves the minimum number of FlexE clients. Therefore, the execution time of FTR is 2x faster than the execution time of RFA. DSP experiences the highest restoration time due to the update delay for rerouting the paths which is the major factor of latency and is fortunately avoidable in FTR and RFA thanks to FlexE supports. We also notice from Fig. 2(b) that the delay increases with the number of failures in FTR, RFA and DSP schemes. Meanwhile, the delay decreases in DSP after $MTTF = 2$, where the performance of DSP leads to more than 50% of traffic loss.

Fig. 2(c) depicts the provisioning cost of PHYs over the number of failures and time, computed as the ratio between the number of new PHYs and the maximum number of available PHYs to recover FlexE clients in each FTR, RFA and MILP (Eq. (13), Eq. (16)). As seen, FTR cost is near to optimal (MILP cost) and lower than RFA cost which stems from the ability of FTR to better utilize the existing resources. In contrast, RFA provisions more PHYs as it prefers adding new PHYs to recover FlexE clients over those subgroups carried over the paths with minimum propagation delay.

V. Conclusion

FTR enables efficient and fast restoration in MLMD networks through tunnel updates, FlexE subgroups reconfiguration and defragmentation. We formulated the FTR problem and derived an approximation mechanism to restore FlexE clients in polynomial time. The proposed mechanism learns off-line the intra-domain latency and the maximum spare capacity that can be used on-line to recover FlexE traffic in a hitless fashion. Experiments in a nation-wide testbed show that FTR recovers more than 97% of traffic, which is higher and faster than the baseline H-PCE restoration. In the future, we will extend our work to scale-up in large fully distributed MLMD networks.

Acknowledgment

The authors thank NSERC, Ciena, and Telus for their supports in the Multi-Layer Multi-Domain Orchestration project.

References

[1] Thyagaturu, Akhilesh S., et al. "Software defined optical networks (SDONs): A comprehensive survey." IEEE Communications Surveys & Tutorials 18.4 (2016): 2738-2786.

[2] Koulougli, Dahina, Kim Khoa Nguyen, and Mohamed Cheriet. "Joint Optimization Of Routing and Flexible Ethernet Assignment In Multi-layer Multi-domain Networks." 2020 29th International Conference on Computer Communications and Networks (ICCCN). IEEE, 2020.

[3] Talebi, Sahar, et al. "Spectrum management techniques for elastic optical networks: A survey." Optical Switching and Networking 13 (2014): 34-48.

[4] Castro, A., et al. "Experimental assessment of bulk path restoration in multi-layer networks using PCE-based global concurrent optimization." Journal of lightwave technology 32.1 (2013): 81-90.

[5] Xu, Feng, et al. "Evaluation of post-fault restoration strategies in multi-domain networks." Optical Switching and Networking 9.2 (2012): 147-155.

[6] Jin, Xin, et al. "Dynamic scheduling of network updates." Proceedings of the 2014 ACM SIGCOMM Conference. 2014.

[7] Xu, Hongli, et al. "Joint route selection and update scheduling for low-latency update in SDNs." IEEE/ACM Transactions on Networking 25.5 (2017): 3073-3087.

[8] The Optical Internetworking Forum. (2019) IA Flex Ethernet 2.1 Implementation Agreement. [Online]. Available: https://www.oiforum.com.

[9] Eira, António, et al. "On the efficiency of flexible ethernet client architectures in optical transport networks." Journal of Optical Communications and Networking 10.1 (2018): A133-A143.

[10] Lu, Wei, et al. "How Much can Flexible Ethernet and Elastic Optical Networking Benefit Mutually?." ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, 2019.

[11] Wang, Rui, and Biswanath Mukherjee. "Provisioning in elastic optical networks with non-disruptive defragmentation." Journal of Lightwave Technology 31.15 (2013): 2491-2500.

[12] Hong, Chi-Yao, et al. "Achieving high utilization with software-driven WAN." Proceedings of the 2013 ACM SIGCOMM Conference. 2013.

[13] Poularakis, Konstantinos, et al. "Learning the optimal synchronization rates in distributed SDN control architectures." IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019.

[14] Jin, Xin, et al. "Optimizing bulk transfers with software-defined optical WAN." Proceedings of the 2016 ACM SIGCOMM Conference. 2016.

[15] Zhang, Mingyang, et al. "Bandwidth defragmentation in dynamic elastic optical networks with minimum traffic disruptions." 2013 IEEE International Conference on Communications (ICC). IEEE, 2013.