Tomography-based progressive network recovery and critical service restoration after massive failures

Viviana Arrigoni, Matteo Prata, Novella Bartolini
Department of Computer Science, Sapienza University of Rome, Italy

IEEE INFOCOM 2023 - IEEE Conference on Computer Communications | 979-8-3503-3414-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOM53939.2023.10228861

***Abstract*—Massive failures in communication networks are a consequence of natural disasters, heavy blackouts, military and cyber attacks. We tackle the problem of minimizing the time and number of interventions to sufficiently restore the communication network so as to support emergency services after large-scale failures. We propose PROTON (Progressive RecOvery and Tomography-based mONitoring), an efficient algorithm for progressive recovery of emergency services. Unlike previous work, assuming centralized routing and complete network observability, PROTON addresses the more realistic scenario in which the network relies on the existing routing protocols, and knowledge of the network state is partial and uncertain. Simulation results carried out on real topologies show that our algorithm outperforms previous solutions in terms of cumulative routed flow, repair costs and recovery time in both static and dynamic failure scenarios.**

***Index Terms*—Network Recovery, Boolean Network Tomography, Massive Network Failure, Flow Routing**

I. Introduction

Communication network outages are frequent events that jeopardize the internet connection, and that can be due to numerous factors, including natural disasters, software attacks and physical attacks [1]. Minor internet outages occur recurrently, as reported, for instance, by the outage mailing list [2] and in the work in [3]. Despite being more sporadic, massive network failures have terrifically disruptive outcomes. Very recently, the Canadian communications and media colossus Rogers went through a massive network outage that also affected ATMs and the 911 service [4]. In 2017, hurricane Irma hit the Caribbeans and caused dozens of victims and a massive failure of communication networks that brought about the isolation of emergency service organizations [5]. Catastrophic events represent a fearsome threat to civilians, administrations and companies. For this reason, recovery is an essential component of the life cycle of the risk management process for all major companies and administrative entities [6], [7], and restoration of communication networks is a lynchpin of the overall recovery plan [8], [9]. One iconic example is the Federal Emergency Management Agency (FEMA) [10], a body of the Department of Homeland Security of the USA that provides emergency assistance and disaster recovery after natural cataclysms. FEMA recognizes the role of the communication network as critical infrastructure. Every government's priority requires collaboration with the business community during emergency, promoting a centralized, holistic approach to emergency management. In emergency scenarios, it is crucial to restore communications to ensure critical services in order to provide critical support. In this paper, we tackle the problem of scheduling node and link repairs in a highly damaged network in order to promptly restore emergency-critical services with the utmost urgency. We formalize this problem as a MILP and prove its NP-hardness. Finding the optimal solution in a realistic setting requires computation times that are prohibitively high. This motivates our proposal of a polynomial algorithm, hereafter called PROTON (**P**rogressive **R**ec**O**very and **T**omography-based m**ON**itoring). PROTON iteratively chooses which network elements to recover in order to restore critical services. The recovery strategy benefits from a monitoring activity that utilizes Boolean Network Tomography techniques based on end-to-end path monitoring. This provides probabilistic information that is crucial for recovery decisions. In contrast to previous approaches to network recovery, PROTON works in realistic settings in which the initial network state is unknown, and information about it can be gained via indirect monitoring or direct inspection. Our approach also works when the monitoring system provides partial information due to the limited accessibility of network elements and routing is decentralized and uncontrollable. In addition, we consider dynamically occurring failures, reflecting disrupting events that may still be ongoing and possibly propagating during the recovery process. By means of a comprehensive experimental campaign, we show that our approach outperforms previous solutions in terms of all the considered performance indicators.

The original contributions of this paper are the following:

- We address the problem of Minimizing the Time for critical Demand Satisfaction, MinTDS, after large-scale failures. We give a MILP formulation of MinTDS and prove its NP-hardness.
- We provide a polynomial algorithm, PROTON, that addresses the problem MinTDS in the realistic scenario in which knowledge of the network state is partial and uncertain. To deal with limited network observability, PROTON integrates probabilistic tomography techniques.
- We carry out a thorough experimental analysis that shows how PROTON outperforms previous solutions, namely CEDAR [11], ShP [12], ISR [13], and other baseline approaches, in terms of all the proposed performance metrics.
- We provide experiments that highlight PROTON's capability to promptly provide suitable recovery decisions to restore critical services even under dynamic failures.

II. Related Work

The increasing dependency of critical infrastructures on the communication network brought about the need to study fast recovery approaches to large-scale network failures. Some works focus on the propagation of failures across interdependent networks and related mitigation strategies, considering the communication network and the power grid [14], [15] or overlay networks [16], [17]. Focusing on the network recovery process as we do in the present paper, the solutions presented in [12], [18] propose a scheduling of repairs of nodes and links in a network that maximizes the weighted sum over time of the flow of every demand pairs. In [19] the authors show a progressive repair strategy that has the goal of minimizing the repair costs. Differently from our contribution, these works assume complete knowledge of the network state. The work in [11] was the first to loosen this assumption: the state of nodes and links is discovered through network monitoring and integrated into a progressive network repairing procedure. The goal of the work in [11] is to maximize cumulative flow between demand nodes. In [13] the authors extend this approach and introduce a scheduling for node and link repairs that aims at minimizing the expected repair costs. The present paper differs from previous works in several aspects. Optimizing for the cumulative flow as in [11], [12], [18], or the recovery cost as in [19], may lead to solutions in which demands are only partially satisfied for several temporal steps. Unlike these works, our objective is to minimize the time for completely satisfying demand flows.

The works [11], [13] assume incomplete knowledge of the network state, but require every node of the network to cooperate in the monitoring activity by means of k-hop ping messages. Differently from these works, the monitoring framework used by our approach relies on a limited set of controllable nodes in the network, whose location may be chosen depending on the type and accessibility of the devices.

III. Notation and Problem Formulation

We model the network as a weighted undirected graph $G = (V, E)$ where V and E are the sets of nodes and links, and each link $(i, j) \in E$ has weight $c_{i,j}$ corresponding to its capacity. We denote with $H = (V_H, E_H)$ the *demand graph*, where $V_H \subset V$ is the set of *demand nodes* that provide critical services and $E_H \subseteq V_H \times V_H$ is the set of *demand edges*, reflecting the required connections between demand nodes. Each demand pair $(s_h, t_h) \in E_H$ has a weight d_h that represents the bandwidth requirement to route its flow with the necessary quality. As a consequence of a disrupting event, some nodes and links of G are non functioning and some or all the demands may not find sufficient bandwidth along paths through G. Hence, V and E are partitioned as $V = \widehat{V}_{\mathtt{W}} \cup \widehat{V}_{\mathtt{B}}$ and $E = \widehat{E}_{\mathtt{W}} \cup \widehat{E}_{\mathtt{B}}$, respectively, where $\widehat{V}_{\mathtt{W}}$ and $\widehat{E}_{\mathtt{W}}$ are the sets of working nodes and links, and $\widehat{V}_{\mathtt{B}}$ and $\widehat{E}_{\mathtt{B}}$ are the sets of broken nodes and links. We consider repair cost $\kappa^l_{i,j}$, for a broken link $(i, j) \in \widehat{E}_{\mathtt{B}}$, and κ^v_i for a broken node $i \in \widehat{V}_{\mathtt{B}}$.

A. MinTDS, Minimum Time for Demand Satisfaction

We initially consider the simplified scenario where the sets of broken nodes and links are known, and formulate MinTDS, i.e., the problem of Minimizing the Time for Demand Satisfaction. MinTDS aims at progressively repairing network elements, within a step-based cost budget (e.g., constraint on daily available personnel for physical inspection of the network components), so that critical demands are fully restored, with minimum average restoration time.

We formalize MinTDS in Equation 1. For each demand pair $(s_h, t_h) \in E_H$ we consider a set of candidate simple paths P^h. $P^h_{i,j}$ is the subset of paths of P^h traversing edge (i, j). We denote with $z^{(n)}_{h,p}$ the fraction of demand flow d_h routed through path p at time n, $z^{(n)}_{h,p} \in [0, 1]$. We introduce $\gamma^{(n)}_h$, a binary decision variable such that $\gamma^{(n)}_h = 0$ if demand d_h is completely satisfied at time n, and $\gamma^{(n)}_h = 1$ otherwise. The relationship between the binary variables $\gamma^{(n)}_h$ and the continuous variables $z^{(n)}_{h,p}$ is expressed by Equation 1b. We want solutions for which $\gamma^{(n)}_h$ is a non increasing function of n for all demands, which implies that once a service demand is restored, it continues to work in the following steps. This is obtained by imposing the constraint of Equation 1c. We define $\delta^{(n)}_{i,j}$ as a binary variable such that $\delta^{(n)}_{i,j} = 1$ if edge (i, j) is repaired at time step n, $\delta^{(n)}_{i,j} = 0$ otherwise, and we define $\delta^{(n)}_i$ analogously for node i. If link $(i, j) \in \widehat{E}_{\mathtt{W}}$ we set $\delta^{(0)}_{i,j} = 1$, and if node $i \in \widehat{V}_{\mathtt{W}}$ we set $\delta^{(0)}_i = 1$. Equation 1d is the link capacity constraint. Equation 1e requires that, for a link to be used at step n, its endpoints must already be working, either from the beginning, or because they have been repaired within step n. With Equation 1f we constraint the cost of repairs to be limited by a budget B_{rep} per time step n. Equation 1g defines the time at which demand h is completely restored.

The objective of MinTDS is to minimise the average time necessary to completely satisfy the demands, that is the same as minimising the sum over all demands h of the variable t_h (Equation 1a). We note that the optimization problem progresses for a number of steps N that is upper bounded by the maximum number of steps $N_{\mathtt{Max}}$ to recover the entire graph, given the budget constraint for each step.

$$\min \textstyle\sum_{h \in E_H} t_h \tag{1a}$$

$$s.t. \quad \forall n = 1, \dots, N$$

$$\gamma^{(n)}_h \geq 1 - \textstyle\sum_{p \in P^h} z^{(n)}_{h,p}, \quad \forall h \in E_H \tag{1b}$$

$$\gamma^{(n-1)}_h \geq \gamma^{(n)}_h, \quad \forall h \in E_H \tag{1c}$$

$$\textstyle\sum_{h \in E_H} \sum_{p \in P^h_{i,j}} z^{(n)}_{h,p} d_h \leq c_{i,j} \sum^n_{k=0} \delta^{(k)}_{i,j}, \quad \forall (i,j) \in E, \tag{1d}$$

$$\textstyle\sum^n_{k=0} \delta^{(k)}_i \geq \delta^{(n)}_{i,j}, \quad \forall i \in V, \forall (i,j) \in E, \tag{1e}$$

$$\textstyle\sum_{(i,j) \in E \setminus E_{\mathtt{W}}} \kappa^l_{i,j} \delta^{(n)}_{i,j} + \sum_{i \in V \setminus V_{\mathtt{W}}} \kappa^v_i \delta^{(n)}_i \leq B_{rep}, \tag{1f}$$

$$t_h = \textstyle\sum^N_{n=0} \gamma^{(n)}_h, \forall h \in E_H \tag{1g}$$

$$z^{(n)}_{h,p} \in [0,1], \gamma^{(n)}_h, \delta^{(n)}_{i,j}, \delta^{(n)}_i \in \{0,1\}, \forall h \in E_H, (i,j) \in E, i \in V. \tag{1h}$$

Theorem III.1. *MinTDS is NP-hard*

Proof. We prove that any instance of the Maximum Coverage (MC) problem, can be reduced to an instance of MinTDS in polynomial time. In MC, we are given a positive integer k, a set of elements E and a collection $\mathcal{S}$ of subsets of E, $\mathcal{S} = \{S_1, \dots, S_m\}$. The objective is to find $\mathcal{S}' \subseteq \mathcal{S}$ such that $|\mathcal{S}'| \leq k$ and the number of covered elements, i.e. elements of E belonging to at least a set in $\mathcal{S}'$, is maximized. The corresponding instance of MinTDS considers a graph G, including a node d. Node d has m neighbors called v_{S_i}, each corresponding to the subset S_i, and connected to d via a broken link of capacity $|S_i|$. For each element $e \in E$, we build a node connected to all nodes v_{S_i} for which $e \in S_i$. We define $E_H = \{(e, d), \forall e \in E\}$, and $d_h = 1$, $\forall h \in E_H$. We assume unitary repair costs for the links $\kappa^l_{i,j} = 1$ and repair budget $B_{rep} = k$. Considering only one repair step, $N = 1$, the constructed instance of MinTDS requires to select the links (d, v_{S_i}) to be repaired, such that the number of satisfied demands is maximized. Given the described construction, this solution corresponds to the optimal solution $\mathcal{S}'$ of MC where each set S_i belongs to the solution $\mathcal{S}'$ if the solution MinTDS suggests to repair link (d, v_{S_i}). Given the polynomial time complexity of the proposed reduction of MC to MinTDS, it follows that MinTDS is at least as hard as MC. $\square$

The hardness of MinTDS motivates us to seek for efficient sub-optimal solutions that can be obtained in polynomial time. The need to work under the realistic scenario of partial knowledge and limited observability of the network state also requires to generalize the problem to integrate network monitoring decisions.

IV. PROTON

In this section, we introduce PROTON (Progressive Recovery and Tomography-based mONitoring), a polynomial time algorithm to solve MinTDS in the more realistic and challenging scenario in which (1) initial damage information is limited and recovery actions must be supported by an efficient progressive monitoring activity based on a limited set of vantage points called *monitors* (i.e., workstations able to initiate and collect measurements.) (2) an uncontrollable default routing algorithm is used to route demands.

In Algorithm 1 we provide the pseudocode of PROTON. The algorithm gets as input the supply and the demand graphs, G and H respectively; the prior failure probability $\pi \in (0,1)$ of nodes and links; a set $\mathcal{M}$ of candidate monitor nodes (i.e., nodes able to initiate and collect measurements), with $V_H \subseteq \mathcal{M}$; a monitor budget $\mu \leq |\mathcal{M}|$ (i.e., a maximum number of monitors deployable on the network); and a positive integer η that is the number of monitors to place at each iteration. All non working demand nodes in V_H are repaired, **line (1)**, and the set of monitor nodes is initialized as the set of demand nodes, **line (2)**. The failure probability for nodes and link is initialized, **line (4)**. The algorithm progresses until all the demands are routable on the supply graph within the network capacity constraints. Each iteration of the while is called a *step*, providing an updated view of the problem instance with additional knowledge and simplified input. In

Algorithm 1 PROTON

```
    Input: $G$ : supply graph; $H$ : demand graph; $\mathcal{M}$: set of candidate monitors; $\mu$: number of monitors; $\pi \in (0,1)$: prior failure probability; $\eta$: number of monitors per iteration
 1: repair all $v \in V_H$
 2: $M \leftarrow V_H$: initialize set of monitors $M$
 3: $n = 0$
 4: $\mathbb{P}^{(n)}[v \in \widehat{V}_{\mathrm{B}} \,|\, v \in M] = 0$, $\mathbb{P}[v \in \widehat{V}_{\mathrm{B}} \,|\, v \notin V \setminus M] = \mathbb{P}[l \in \widehat{E}_{\mathrm{B}}] = \pi$
 5: while $H^{(n)}$ is not routable on $G^{(n)}$ do
 6:     $n \leftarrow n + 1$
        ------------------------------------------------> monitoring
 7:     $\mathbb{P}^{(n)}, \mathcal{O}^{(n)} \leftarrow$ MONITORING $(\mathbb{P}^{(n-1)}, H^{(n-1)}, \pi)$
        ------------------------------------------------> path pruning
 8:     $\mathcal{P}_{\mathtt{W}} \leftarrow \{p \,|\, p \in P^{h(n)} \wedge \mathcal{O}^{(n)}(p) = 0, \forall h \in E_H^{(n)}\}$
 9:     prune all $h$ over $p \in \mathcal{P}_{\mathtt{W}}$ by decreasing priority $\psi(h, p)$    (Eq. 2)
        and update $H^{(n)}$ accordingly
        ------------------------------------------------> repair
10:     $\mathcal{P}_{\mathrm{R}} \leftarrow \emptyset$
11:     for $(s_h, t_h) \in E_H^{(n)} \,|\, \nexists p \in P^{h(n)} \wedge \mathcal{O}^{(n)}(p) = 0$ do
12:         $p_h^* = \arg\min_{p \in P^{h(n)}} l^{(n)}(p)$    (Eq. 6)
13:         $\mathcal{P}_{\mathrm{R}} \leftarrow \mathcal{P}_{\mathrm{R}} \cup \{ p_h^* \}$
14:     select $\bar{p} \leftarrow \arg\min_{p \in \mathcal{P}_{\mathrm{R}}} \ell^{(n)}(p)$
15:     repair $\{\widehat{V}_{\mathrm{B}}^{(n)} \cap V_{\bar{p}}\} \cup \{\widehat{E}_{\mathrm{B}}^{(n)} \cap E_{\bar{p}}\}$
16:     $\mathbb{P}[v \in \widehat{V}_{\mathrm{B}} \,|\, v \in V_{\bar{p}}] = 0$, $\mathbb{P}[l \in \widehat{E}_{\mathrm{B}} \,|\, l \in E_{\bar{p}}] = 0$
        ------------------------------------------------> monitor placement
17:     for $1, \dots, \eta$ do
18:         if $|M| < \mu$ then $v \leftarrow \arg\max_{v \in (\mathcal{M} \cap V_{\mathtt{W}}^{(n)}) \setminus M} u(v)$    (Eq. 8)
19:             $M \leftarrow M \cup \{v\}$
```

the following, the updated view of the problem instance at a given step is denoted with the time step index (n). Each step is divided into four main *phases*:

1) Monitoring: The goal of the monitoring step is to infer network nodes and links' state, *working* or *failed*.

Due to limitations on the set of nodes that can act as monitors and on the monitor budget μ, we resort to Boolean Network Tomography techniques, as explained in detail in Section IV-A. The idea of performing monitoring incrementally, along with the recovery process, is to exploit the potentially restored connectivity after each repair to obtain measurements involving larger portions of the network.

The monitoring phase, **line (7)**, returns $\mathbb{P}^{(n)}$, that is the failure probability distribution over nodes and links, and $\mathcal{O}^{(n)}$, that is the binary outcome of the monitored paths. In particular, it provides $\mathbb{P}^{(n)}[v] = \mathbb{P}[v \in \widehat{V}_{\mathrm{B}}^{(n)} | \mathcal{O}^{(n)}]$, i.e., the conditional probability of failure of a node v, based on observations, and the analogous evaluation for the link failure probability, $\mathbb{P}^{(n)}[l] = \mathbb{P}[l \in \widehat{E}_{\mathrm{B}}^{(n)} | \mathcal{O}^{(n)}]$. At each step n, the distribution $\mathbb{P}^{(n)}$ is updated and used for selecting the elements to repair.

2) Path pruning: After the monitoring phase, for all the demands, we can tell if the nodes of a demand pair $(s_h, t_h) \in E_H^{(n)}$ can communicate along a path $p \in P^{h(n)}$, which is the set of working paths connecting demand pairs yet to satisfy and determined by the default routing protocol. In such a case, **line (8)**, we can update the problem instance reducing both the demand flow and the links' capacity along p accordingly, considering the residual capacity. Let $c_p^{(n)} = \min_{(i,j) \in p} c_{ij}^{(n)}$ be the capacity of path p at time n. The action of pruning $d_h^{(n)}$ over p is a simplification of the problem instance obtained by reducing the demand

$d_h^{(n)}$ and the capacity of the links of p of an amount equal to $d = min\{c_p^{(n)}, d_h^{(n)}\}$, so that $d_h^{(n+1)} = d_h^{(n)} - d$ and $c_{ij}^{(n+1)} = c_{ij}^{(n)} - d$, for all $(i,j) \in p$. Demand pairs whose flow requirement $d_h^{(n)} = 0$ becomes null as a consequence of a pruning action, are considered satisfied.

We note that the pruning phase may encounter conflicting decisions when multiple demands can be pruned along paths sharing common links. In fact, the order in which demands are pruned plays a relevant role in the *feasibility* of the solution, i.e., in the ability of the algorithm to route the entire demands. For this reason, we sort conflicting demands with respect to a *priority* as follows. Let $p \in P^h$, with capacity $c_p^{(n)}$, be a working path connecting s_h and t_h at time step n, in which the related flow demand is $d_h^{(n)}$. Let $\mathcal{P}_{\mathtt{W}}^{(n)}$ be the set of prunable paths returned at step n. If multiple demands have a conflict on paths in $\mathcal{P}_{\mathtt{W}}^{(n)}$, sharing one or more common links, conflicting demands are pruned in decreasing order of their priority values:

$$\psi(h,p)^{(n)} = d_h^{(n)}/c_p^{(n)}. \tag{2}$$

We then prune such demands (if any), starting from the ones with the highest priority, **line (9)**. This allows to prune paths that serve high demands and have small capacities first, in order to prevent unfeasibility.

3) Repair: For every couple of demand nodes (s_h, t_h) that cannot communicate (i.e., no working path between them exists, **line (11)**), we select a path p_h^* according to the criterion of minimum stochastic path length, **lines (12)-(13)** (the motivation and detailed explanation of this criterion are provided in a specifically devoted Section IV-B, for clarity of presentation). Among all such demand pairs, we select the one connected by the path with the lowest length, and repair the broken components of such path, **lines (14-15)**, where V_p and E_p are the set of nodes and links, respectively, traversed by path p. We update the failure probability distribution accordingly **line (16)**, i.e., $\forall v \in V_p$ and $\forall l \in E_p$ such that p is the path selected in line (14), we set $\mathbb{P}^{(n)}[v] = \mathbb{P}^{(n)}[l] = 0$.

4) Monitor placement: If the budget on the maximum number of monitors μ has not been reached yet, we place a number of monitors per iteration η, as detailed in Section IV-D. This operation allows to create more end-to-end monitoring paths, which provide a finer estimation of the failure probability of nodes and links.

A. Network Tomography and Failure Probability Estimation

Existing recovery algorithms work under the unrealistic assumption that working and failed nodes/links are known (e.g., [12], [19], [16]) or that knowledge of the state of nodes and links can be gained easily, as all working nodes are potentially monitors and responsive to ping messages (e.g., [11], [13]). While the first hypothesis is unrealistic, the latter raises two major issues. First, it is unlikely to assume that all nodes in the network can be employed as monitors; in fact, many nodes are non-programmable layer 2 and 3 devices that cannot possibly run monitoring software. Second, deploying a monitor has a cost (in terms of time, hardware and equipment expenses) that existing solutions do not account for. In contrast to previous work, we relax the aforementioned assumptions and use monitoring techniques based on network tomography to gain knowledge of the state of the network. Boolean Network Tomography (BNT) [20] is a powerful tool for gaining knowledge of the state of internal nodes and links of a network by sending monitoring probes between endpoints located at the periphery of the network. This technique overcomes typical issues in other network monitoring technologies that are due to network heterogeneity. Furthermore, it limits expensive and risky human interventions in potentially dangerous settings. In BNT, the state of the nodes and links is assumed to be binary (working/failed). Accordingly, a path between two endpoints is working if it only traverses working nodes and links, and failed otherwise. Network tomography exploits information gained by end-to-end monitoring messages between monitor nodes to gain knowledge on the state of traversed nodes and links as follows. Given a couple of monitor nodes v_1, v_2, we send a monitoring packet between them. If a working path between the two nodes exists, we can infer the state of all nodes and links traversed by such path as working. In this case, the route of the path is decided by the underlying routing algorithm. If nodes v_1 and v_2 cannot communicate with one other, we know that every possible path connecting v_1 and v_2 traverses at least a failed node or link. Based on the binary outcome of path probes between monitors $\mathcal{O}^{(n)}$, we can infer the probability of failure of a node v as $\mathbb{P}[v \in \widehat{V}_{\mathtt{B}}^{(n)} | \mathcal{O}^{(n)}]$ and analogously for a link l, $\mathbb{P}[l \in \widehat{E}_{\mathtt{B}}^{(n)} | \mathcal{O}^{(n)}]$. We let $V_{\mathtt{W}}^{(n)} \subseteq \widehat{V}_{\mathtt{W}}^{(n)}$ be the set of nodes that we know being working, and $V_{\mathtt{B}}^{(n)} \subseteq \widehat{V}_{\mathtt{B}}^{(n)}$ the set of nodes that we know being broken, i.e., $V_{\mathtt{W}}^{(n)} = \{v \in V : \mathbb{P}[v \in \widehat{V}_{\mathtt{B}}^{(n)} | \mathcal{O}^{(n)}] = 0\}$, $V_{\mathtt{B}}^{(n)} = \{v \in V : \mathbb{P}[v \in \widehat{V}_{\mathtt{B}}^{(n)} | \mathcal{O}^{(n)}] = 1\}$. We define sets of edges $E_{\mathtt{W}}^{(n)}$ and $E_{\mathtt{B}}^{(n)}$ analogously. It results that the joint probability of the path outcomes $\mathcal{O}^{(n)}$ is:

$$\mathbb{P}[\mathcal{O}^{(n)}] = 1 + \sum_{k=1}^{|\mathcal{F}^{(n)}|} \left((-1)^k \sum_{\substack{S \in 2^{\mathcal{F}^{(n)}}: \\ |S|=k}} (1-\pi)^{\left| \bigcup_{p\in S} (V_p \setminus V_{\mathtt{W}}^{(n)}) \cup (E_p \setminus E_{\mathtt{W}}^{(n)}) \right|} \right), \tag{3}$$

where $\mathcal{F}^{(n)}$ is the set of failed paths at step n, $2^{\mathcal{F}^{(n)}}$ is the power set of $\mathcal{F}^{(n)}$, and π is a priori estimate of the failure probability of nodes and links. Therefore, the conditional probability of failure of a node v is computed as follows:

$$\mathbb{P}[v \in \widehat{V}_{\mathtt{B}}^{(n)} | \mathcal{O}^{(n)}] = \pi \cdot \mathbb{P}\left[\bigwedge_{\substack{p \in \mathcal{F}^{(n)} \\ v \notin p}} p \in \mathcal{F}^{(n)} \right] / \mathbb{P}[\mathcal{O}^{(n)}]. \tag{4}$$

The conditional probability of failure of a link is defined analogously. Notice that these equations require a number of operations that is exponential in the number of failed paths. In case of massive failures, such operations are intractable. For this reason, we adopt the same solution used in [21] to approximate conditional failure probabilities based on end-to-end measurements via a *failure centrality* metric. Failure centrality approximates failure probabilities and is computable in polynomial time. In particular, we slightly modify the definition of failure centrality in [21] to be dependent also on the prior failure probability π.

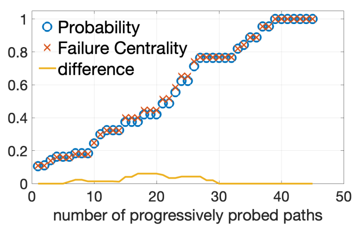


Fig. 1: Progression of $\mathbb{P}[v \in \widehat{V}_{\mathtt{B}} | \mathcal{O}]$ and $c(v|\mathcal{O})$, $k = 10$.

Definition IV.1. *The failure centrality of a node v based on the observations on the path probes $\mathcal{O}^{(n)}$, $c(v|\mathcal{O}^{(n)})$, is equal to the prior failure probability π if v is not traversed by any failing path and is equal to 0 if it is traversed by a working path or if it is repaired. Otherwise, it is given by $c(v|\mathcal{O}^{(n)}) = \max\{T_1; T_2\}$, where:*

$$T_1(v|\mathcal{O}^{(n)}) = \left\lceil \frac{1}{|\mathcal{F}_v|} \cdot \sum_{p \in \mathcal{F}_v} \left\lfloor \frac{1}{|p \setminus (V_{\mathtt{W}}^{(n)} \cup E_{\mathtt{W}}^{(n)})|} \right\rfloor \right\rceil, \tag{5a}$$

$$T_2(v|\mathcal{O}^{(n)}) = \pi + (1-\pi)\left(1 - \frac{1}{(\mathcal{R}_v + 1)^{k\pi}}\right) \tag{5b}$$

$$\mathcal{R}_v(v|\mathcal{O}^{(n)}) = |\mathcal{F}_v| / |\bigcup_{p \in \mathcal{F}_v} p \setminus (V_{\mathtt{W}}^{(n)} \cup E_{\mathtt{W}}^{(n)})|, \tag{5c}$$

being $\mathcal{F}_v$ the set of failing paths traversing v and $k > 0$ is a tuning parameter.

The centrality of a link l, $c(l|\mathcal{O}^{(n)})$, is defined analogously. Nodes and links' failure centrality lies in $[0, 1]$, in analogy with failure probability. Notice that $T_1 \in \{0, 1\}$. The work in [21] proves that if $\max\{T_1, T_2\} = T_1(v|\mathcal{O}^{(n)}) = 1$, then node v is failed. When $T_1 = 0$, the state of the corresponding node or link is uncertain, and its failure probability is approximated by $T_2 = \max\{T_1, T_2\} < 1$. The definition of failure centrality relies on the observation that the failure probability of a node/link is directly proportional to the number of failing paths traversing it and decreases with the length of such paths, as Equations 3 and 4 suggest. The term $\mathcal{R}_v$ in Equation 5c (which is always ≥ 0) captures this trend, which is preserved by T_2, that has the role of mapping $\mathcal{R}_v$ in the interval $[\pi, 1)$.

In Figure 1, with an experiment, we show the curves representing the progressive values of $\mathbb{P}[v \in \widehat{V}_{\mathtt{B}} | \mathcal{O}]$ and $c(v|\mathcal{O})$ as more and more paths are being probed, where v is a failed node. Because of the high computational complexity of evaluating the exact failure probability, we ran this experiment on a small randomly generated network of 27 nodes. The number of monitors is 10; hence there are 45 possible paths to probe. The curves are averaged on 40 runs. At each run, five different nodes fail. The maximum difference between the two curves is 0.061, and the average is 0.0178, showing that the failure centrality is a good approximation of the exact failure probability.

Proposition IV.1. *The computational cost of failure centrality is $O(\mu^2 \cdot l_{\mathtt{Max}})$, where $l_{\mathtt{Max}}$ is the maximum path length.*

Proof. The most expensive operation in Equation 5 is computing $|\cup_{p \in \mathcal{F}_v} p \setminus (V_{\mathtt{W}}^{(n)} \cup E_{\mathtt{W}}^{(n)})|$. It requires counting the number of elements that are not in $V_{\mathtt{W}}^{(n)} \cup E_{\mathtt{W}}^{(n)}$ for all possible failed paths traversing v. All such paths are at most $(\mu-1)(\mu-2)/2$, and their maximum path length is $l_{\mathtt{Max}}$. ∎

In our MONITORING routine, we implement the FaCe-Greedy algorithm introduced in [21]. FaCeGreedy aims to infer the maximum knowledge on nodes and links' state with the minimum number of path probes, that is, probing only *useful* paths between monitors, instead of all monitoring paths. This algorithm returns nodes and links' failure probabilities. A path probe is as useful as it contributes to incrementing the knowledge of nodes and links' state. In this sense, probing a path whose elements are a subset of already working paths such that their links have positive residual capacity does not provide additional information about the state of the network. Similarly, if no working path between two monitors v_1, v_2 exists, then any monitor v_1' connected to v_1 by a working path is not going to be able to communicate with any monitor v_2', which is in turn connected to v_2 by a working path. Such end-to-end paths have zero utility, as probing them would not provide any additional information on the state of the network. Avoiding probing zero utility paths reduces the time spent on monitoring and the amount of injected traffic.

B. Selection of a path to repair

For each couple of demanding edges $(s_h, t_h) \in E_H^{(n)}$ whose communication is disrupted, we select a set of nodes and links to repair that establishes a path between s_h and t_h. We introduce the notion of *stochastic path length* that grounds in the observations provided by end-to-end monitoring paths to guide this decision:

$$\ell^{(n)}(p) = \begin{cases} \ell_{\mathbb{P}}^{(n)}(p) \text{ if } |p|_{\mathtt{hops}} \leq l_{\mathtt{Max}} \\ \infty \text{ otherwise} \end{cases} \tag{6}$$

where with $|p|_{\mathtt{hops}}$ we refer to the number of hops in p, and $l_{\mathtt{Max}}$ is an upper bound on the hop count of used paths introduced for QoS requirements. The value of $\ell_{\mathbb{P}}^{(n)}(p)$ is the following:

$$\begin{aligned} \ell_{\mathbb{P}}^{(n)}(p) = &\sum_{i \in V_p} \kappa_i^v \cdot \mathbb{P}[i \in \widehat{V}_{\mathtt{B}}^{(n)} | \mathcal{O}^{(n)}] + \\ &\frac{1}{c_p^{(n)}} \sum_{(i,j) \in E_p} \kappa_{(i,j)}^e \cdot \mathbb{P}[(i,j) \in \widehat{E}_{\mathtt{B}}^{(n)} | \mathcal{O}^{(n)}]. \end{aligned} \tag{7}$$

with V_p, E_p being the nodes and links traversed by path p.

The first term of $\ell_{\mathbb{P}}^{(n)}(p)$ is the expected value of the cost of repairing nodes traversed by p. The second term is the expected value of the cost of repairing edges traversed by p, divided by $c_p^{(n)}$, i.e., the residual capacity of path p at step n. Driven by this metric, we favour the repair of paths with the minimal expected cost of repair and with large residual capacity, i.e. the path of minimum stochastic length $\ell^{(n)}(p)$.

As we anticipated in Section IV-3, the repair of a path might imply that the corresponding demand is routed on it. In the examples of Figure 2, v_1 and v_2 are demand nodes such that $(v_1, v_2) \in E_H^{(n)}$. Dark nodes are broken, light nodes are working, and square nodes are demand nodes. Nodes are labelled with their failure probability. In the case of Figures 2a and 2b, the path being repaired is also the one through which the demand is routed (either because it is the only working

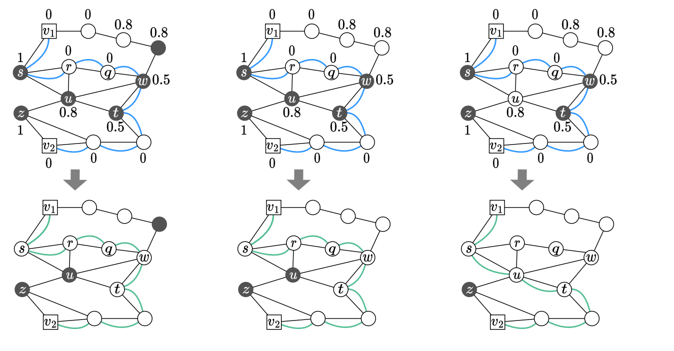


Fig. 2: Three different routing outcomes. Blue paths in top lines are $p = \arg\min \ell^{(n)}(p)$. Green paths in bottom line are used by the routing algorithm to route demand flows after repair of nodes s, t, w.

path between the endpoints, as in Figure 2a, or because it is the one of minimum weight for the routing algorithm, as in Figure 2b). In the case of Figure 2c, the routing algorithm uses the repaired path only partially. To prevent potentially useless repairs, we progressively send monitoring packets between the corresponding demand pair after each repair of the path.

Proposition IV.2. *The computational complexity of $\ell^{(n)}(p)$ is $O(|E| + |V| \log |V| + \mu^2 \cdot l_{Max} \cdot (|V| + |E|))$, where l_{Max} is the maximum path length.*

Proof. Computing $\ell^{(n)}(p)$ has the cost of the Dijkstra algorithm ($\Theta(|E| + |V| \log |V|)$) and of computing the failure centrality for all nodes and links (Proposition IV.1). ∎

C. Routing algorithm

Previous works on progressive network recovery [11], [19], [12] formalize optimization problems to route demand flows and solve them either by standard relaxation methods or by heuristic algorithms. The resulting solutions usually provide flow variables that regulate the amount of traffic passing through every link (i, j), and that consequently define routing paths. Such solutions can be extremely convenient in theory, but they overlook the routing algorithms that network routers implement. Hence, the routers might not be able to utilize such solutions in practice. In this work, we account for this fundamental factor, and we route flows relying on the underlying routing algorithm implemented in the network devices.

D. Monitor Placement

Boolean Tomography provides probing of end-to-end paths to assess the damage of the network components according to the binary outcome of such tests. In the considered scenario, we assume to have a budget μ for the number of monitors deployable in the network. We assume that every demand node acts as a monitor node, and therefore $\mu \geq |V_H|$. If $\mu > |V_H|$, a number of additional monitor nodes can be deployed to increase network knowledge (line (18) in Algorithm 1).

Notice that the problem of optimal monitor placement for network tomography-based monitoring is NP-hard [22]. Hence we adopt a utility-based heuristic approach.

In particular, we want to place the most useful monitor $v_m \in \mathcal{M} \setminus M$. A monitor is as useful as it contributes to gaining knowledge of the state of the nodes and links in the network. The utility of placing a new monitor v_m depends on the utility of the new paths that have an end-point in v_m, that in turn is proportional to the number of new nodes and links' state that monitoring such paths would allow discovering. Such operation requires evaluating the failure probability of all nodes and links traversed by all monitoring paths p between v_m and v, $\forall v_m \in \mathcal{M} \setminus M$ and $\forall v \in M$, as in Equation 4, for all possible outcomes of such path probes. As discussed in Section IV-A, this is also computationally intractable. For each possible monitorable node $v_m \in \mathcal{M} \setminus M$, we define the following approximate utility metric:

$$u(v_m) = |\{p : p \in P^{h(n)}, v_m \in p = \arg\min_p \ell^{(n)}(p), h \in E_H^{(n)}\}|. \quad (8)$$

The utility $u(v_m)$ corresponds to the number of shortest paths (with respect to the path length defined in Equation 7) that serve still unsatisfied demands and traverse v_m.

Proposition IV.3. *The computational cost of Equation 8, assuming that $\ell^{(n)}(p)$, $p \in P^{h(n)}$, is already known $\forall h \in E_H^{(n)}$, is $O(l_{Max} \cdot |E_H^{(n)}|)$.*

Proof. To compute the utility of v_m, we need to scroll each path (whose length is upper-bounded by the maximum path length l_{Max}) for all possible residual demands. ∎

V. Performance evaluation

In this section, we evaluate the performance of PRoTOn by means of simulations and compare it with previous solutions and baseline approaches. We first describe the simulation environment and related metrics.

Simulator. The experiments described in this section were carried out on NetMARS (**Net**work **M**onitoring **a**nd **R**ecovery **S**imulator), a simulation framework including libraries to visualize and process real computer network topologies, simulate disruption scenarios, implement damage assessment strategies providing probabilistic and partial knowledge, recovery algorithms, and commonly used routing protocols. We made NetMARS available for download [23] to ensure the reproducibility of our analysis.

Computer Network Topology and Routing. As a real network topology for the experiments, we consider the fibre network of Minnesota. This topology results from the collaboration of more than 50 carriers in the state. Its dataset was made available by Aurora Fiber Optic Network and is integrated with the NetMARS downloadable dataset [23]. It consists of 681 nodes and 921 edges. We define the link capacity in terms of flow units. Backbone links are set to 160 flow units, while the other links are set to 80 flow units.

We consider multi-path load-sensitive routing protocols [24]. Although we do not make assumptions regarding specific routing algorithms and protocols, in our simulations we implement a routing algorithm inspired by the EIGRP protocol standardized by Cisco, that searches for the paths of minimum weight that connect the demand endpoints, as defined in [25].

Simulated Scenarios. In the experiments we consider the impact of 4 independent variables on the performance of PROTON. (i) *Network Disruption (%)* represents the percentage of broken elements in the network. Network Disruption is simulated as a bi-variate Gaussian distribution, of which the standard deviation models the extent of the disruption. When not otherwise stated, we consider a unique disruption epicentre and vary the failure extent within the range 30%-80% of nodes and edges of the network. (ii) *Number of Demand Pairs* represents the number of edges in the demand graph (i.e. $|E_H|$), assuming each has uniform flow requirement d_h. (iii) *Demand Flow* represents d_h, that is the quantity of flow each demand pair needs to route from source to destination. We assume homogeneous demands, whose flow requirement varies from 10 to 30. (iv) *Number of monitors* represents the upper-bound μ on the number of monitors.

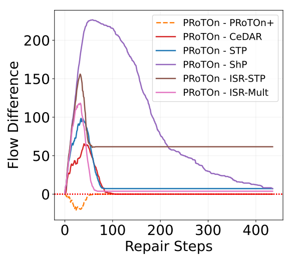
Key Performance Indicators (KPIs). We measure the performance of the analyzed algorithms in terms of: (i) *Routed Flow:* it is the total quantity of demand flow routed until the i-th repair step. It includes partially routed flows. (ii) *Cumulative Flow:* it is the integral of the routed flow over time. It is a measure of early demand restoration. A higher cumulative flow means that the repair strategy prioritizes early service restoration. Even two recovery plans providing the same set of repairs but different schedules in time may end up in remarkably different cumulative flow values. (iii) *Total Flow:* it is the sum of the flows of the entirely routed demands at the end of the recovery process. (iv) *Cumulative Restoration Time:* it represents the sum of the restoration time of the demands. This metric is of key relevance, and it appears in the objective function of Equation 1a. (v) *Number of Repairs:* it represents the number of repairs proposed by the recovery plan until the end of the recovery process. It includes both edges and nodes. Results are randomized on 80 runs, in which the position of the epicentre of the disruption and the position of the demand endpoints vary. In all the plots, the error bars represent the standard error. In the experiments, we assume that the initial state of the network elements is unknown and can be updated based on the monitoring activity which precedes any repair decision.

A. State of the Art Comparison

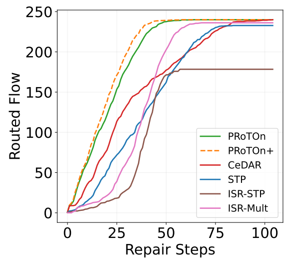
We compare the performance of PROTON with respect to previous literature solutions and baseline approaches.

With CEDAR and ISR, the initial state of nodes and links is unknown. To update the state of nodes and edges, these methods implement a *k-hop monitoring* strategy. It consists in placing a monitor on the repaired nodes, from where to ping towards the nodes at most k hops away. We underline that this is a very strong assumption, as it implies that all the network nodes are collaborative and respond to the monitors as requested.

ISR [13] aims at minimizing the cost of repairs. It is proposed in two variants, ISR-STP selects the paths to repair based on a weighted shortest path approach, whereas ISR-Mult solves a relaxed multi-commodity linear program through budget-constrained iterative steps to determine which components to repair. Both ISR-STP and ISR-Mult may not be able to route the entire demand flow due to possible violations of the link capacity constraints [13]. CEDAR [11] is an algorithm for network recovery and monitoring with the goal to maximize the cumulative flow routed over time. It relies on the unrealistic assumption that an unbounded number of monitors can be placed anywhere in the network. In fact, CEDAR proposes to repair only paths whose components' states are completely known; otherwise the algorithm loops waiting for additional information from the monitoring system. To keep the comparison fair, we allow CEDAR to randomly guess the state of the network elements rather than prematurely stopping the algorithm once it exceeds the budget of monitors. We also run comparisons with ShP [12], an algorithm for network recovery whose objective is to maximize the weighted cumulative flow. ShP does not handle partial knowledge of the state of the network. To run fair comparisons, we let also ShP run $k-$hop monitoring. Differently from PROTON, all these methods assume routing to be fully controllable.



(a) Difference of routed flow.



(b) Routed flow.

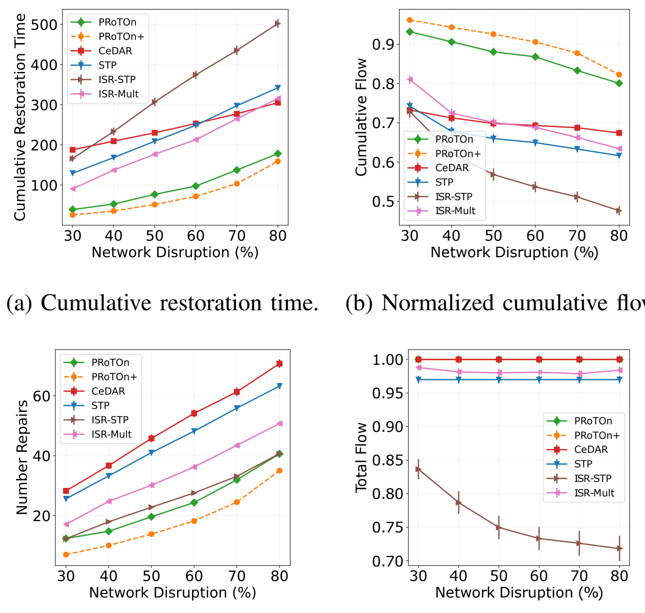
Fig. 3: Routed flow over time.

Finally, we run comparisons with two baselines. One is PRoTOn+, that is PROTON under complete knowledge of node and link state. In the static failure scenario considered in this section, the nodes and links that are already working or have been repaired never fail during the recovery process. For this reason, PRoTOn+ does not perform monitoring. The other baseline is STP which progressively repairs the shortest paths (hop-based) between demand endpoints, gradually routing the restored demands and working with the residual capacity supply graph. STP does not use knowledge of the network state in making repair decisions; hence it does not perform monitoring. Similar to PROTON, for PRoTOn+ and STP, the flow is routed according to the IP algorithm described in Section IV-C (i.e., paths containing failed nodes or links or 0-capacity links are avoided).

B. Simulation Results

In all the experiments we set a unitary repair budget at each step; thus, the repair steps are mapped to time. Furthermore, the network disruption is set to 80%, $|V_H| = |E_H| = 8$, $d_h = 30\ \forall h$ and $\mu = 20$.

We first evaluate the algorithms in terms of normalized routed flow during the recovery process. Figure 3 shows an

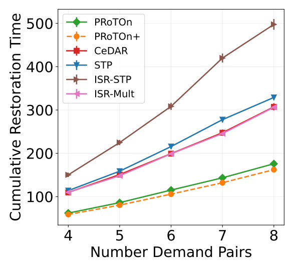


(a) Cumulative restoration time. (b) Normalized cumulative flow.

(c) Number of repairs. (d) Normalized total flow.

Fig. 4: Varying percentage of broken nodes and links.

average calculated over 80 runs, from time 0 to the time at which all the algorithms reach their maximum routed flow. In particular, Figure 3a shows the flow difference between the flow routed by PRoTOn and all other methods, whereas Figure 3b shows the curves of the progressively routed flow for all methods. Figure 3a uses a larger timescale than Figure 3b, as it also includes the performance of ShP. This experiment demonstrates that ShP is not directly comparable with the other algorithms. We recall that ShP maximizes the weighted total flow, thus it does not show any effort in minimizing the number of repairs done. For this reason, we do not include ShP in other plots. PRoTOn+ constitutes a theoretical limit to the performance of PRoTOn, as it is the same algorithm which makes decisions on the basis of perfect knowledge of the network state. We notice that the two algorithms perform closely. PRoTOn requires 17% more repair steps than PRoTOn+ to route the entire demand flow. Such a close distance, also highlighted in Figure 3a, underlines the high accuracy of tomography in network state estimation. The figure captures the advantages of PRoTOn. First, we notice that it takes on average 65 repair steps to restore the demands fully, that is 54% fewer repairs with respect to CeDAR, which is the second best, routing the whole flow with 104 repairs. We further notice that ISR-STP routes at most 74% of the flow, with 58 repairs, when for PRoTOn 30 repairs are sufficient to restore the same amount of flow, that is 93% fewer repairs. In Figure 3a, it is also evident how PRoTOn stands out for early restoration as the pairwise flow difference between PRoTOn and the other algorithms is consistently positive over time. Figure 4 shows the other KPIs when we vary the percentage of broken network elements from 30% to 80%. Both the cumulative restoration time of Figure 4a and the cumulative flow of Figure 4b reflect the algorithms' capability to provide early service recovery. PRoTOn+, i.e., the complete knowledge variant of our approach, improves the state-of-the-art counterparts between 14% and 22%, whereas the improvement of PRoTOn ranges between 14% and 19%. Although CeDAR's goal is to optimize the cumulative flow over time, PRoTOn improves CeDAR's performance by up to 18%. This is only partially due to the monitor budget, in fact, Figure 3b shows that even within the very first time steps, the growth of CeDAR is noticeably slower than PRoTOn's, resulting in a smaller cumulative flow. Figure 4a represents the cumulative time for demand restoration (Equation 1a) which understandably increases as more and more network elements are damaged. Notice that all methods require at least double the time needed by PRoTOn for routing all the demand flows. Figure 4c represents the average number of repairs required by all the considered methods to converge, which grows linearly as the network disruption grows. Although ISR-Mult and ISR-STP optimize the expected repair costs, they require more repairs than PRoTOn in order to converge to solutions that may not route the entire flow, as Figure 4d proves. These results show how PRoTOn is able to make repair decisions that favour the early total recovery of demand flows whilst minimizing repair costs for different percentages of network disruption. Figure 4d shows that no matter the effort made by ISR-Mult, ISR-STP and STP in minimizing repairs, they cannot route the whole flow. This is because of the unfeasible set of recovered paths produced by their recovery plans due to their violations of link capacity constraints. The difference between PRoTOn which routes the whole flow, and ISR-STP routing between 17% and 30% less flow when the network disruption varies, is even more remarkable than with the other algorithms. Figures 5a and 5b illustrate the cumulative restoration time and the number of repairs when the number of demand edges in $|E_H|$ increases from 4 to 8. The plot highlights how the distance between PRoTOn and the other algorithms increases as the number of service demands increases, with all demands being fully restored 110 time units (+60%) earlier on average, with respect to the second best result, when the number of demands is 8. These results highlight a considerable improvement with PRoTOn in early service recovery. Figures 6a and 6b illustrate the cumulative restoration time and the number of repairs when the demand flow for all the demand pairs $|E_H| = 8$ increases from 10 to 30. It is worth noting how, in figure 6b, the number of repairs of STP and ISR-STP is independent of the increase in the demand flow. This is due to the fact that their decision-making does not take into account the capacity of the demand edges. Finally, in Figure 7, we evaluate the cumulative restoration time and the number of repairs when varying the number of available monitors. The algorithms that do not perform monitoring are omitted from this experiment. We can see that, in terms of time for restoration (Figure 7a) and number of repairs (Figure 7b), all methods benefit from a higher monitor budget, except for ISR-Mult, whose performance is about unchanged. The wider availability of monitors is particularly convenient for CeDAR, which is extremely dependent on its


(a) Cumulative restoration time.

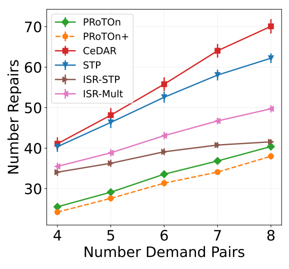
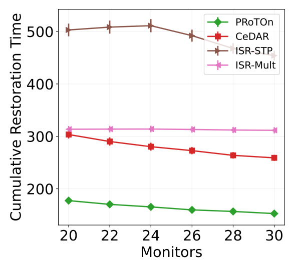

(b) Number of repairs.

Fig. 5: Varying number of service demands.


(a) Cumulative Restoration Time

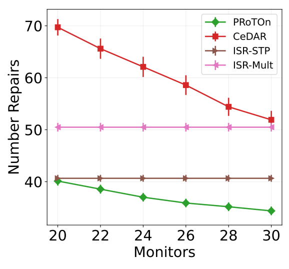
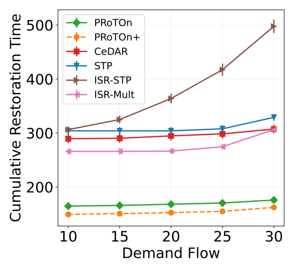

(b) Number of repairs

Fig. 7: Varying number of monitors.


(a) Cumulative restoration time

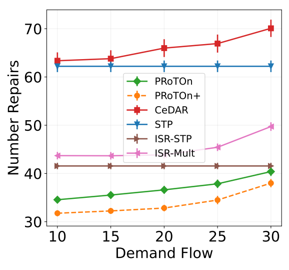

(b) Number of repairs.

Fig. 6: Varying flow requirement per service demand.

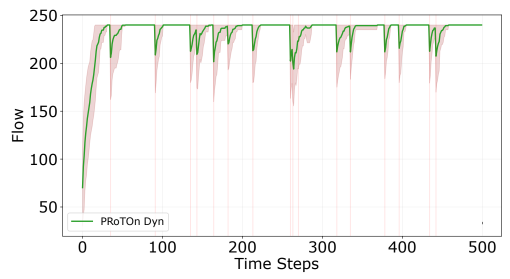


Fig. 8: Routed flow in a dynamic disruption scenario.

ability to localize failed nodes and links with certainty. We highlight that for the settings represented in Figures 5, 6 and 7, only PRoTOn, PRoTOn+ and CeDAR always route the entire demand flow, resembling the trend shown in Figure 4d.

Our experiments show that PRoTOn outperforms all other methods in all the considered network scenarios, routing more service demands earlier, with fewer repairs and more efficient use of the monitor budget, thanks to the integration of tomographic monitoring techniques. PRoTOn proves to be superior to the other algorithms also because of its ability to better cope with partial and uncertain knowledge of the failed network components.

C. Dynamic failures

This section shows how PRoTOn can be extended to deal with dynamic network failures. Natural disasters and malicious attacks can cause a cascading effect that triggers a failure propagation process, which is typically not instantaneous [26], [27], [5]. For this reason, some nodes and links might fail even throughout the recovery intervention process. To tackle this problem, we assume that nodes and links that are originally working might fail abruptly, even after some demand flows have been routed through them. This event possibly causes losses of connection between demand nodes. We also assume that repaired nodes and links are made robust against this cascading effect and that it is less likely that they break again. In the next experiments, we model node and link failures as a Poisson process of rate $\lambda = 0.03$ failures/time steps. When a working node fails, it may ignite a cascade event that makes its neighbours fail with a certain probability, modelled as a logistic function, as suggested in [28]. To handle the dynamics of cascading failures, we modify PRoTOn as follows: (1) we remove the termination condition in line (5), (2) we probe all monitor pairs at each iteration instead of monitoring only *useful* end-to-end paths, as described in Section IV-A, moreover (3) we sporadically send probe packets even when all service demands have been restored, to detect possible new failures. In Figure 8 we show how the routed flow varies with the progression of time steps under this dynamic failure scenario. Red vertical lines represent time steps in which independent failures occur, each provoking a cascading failure effect. Shades represent the standard deviation measured over 100 runs. This experiment highlights how PRoTOn can be easily adapted to work efficiently even in the presence of dynamic failures.

VI. Conclusions

We consider the problem of joint damage assessment and progressive recovery of a communication network after a massive failure under incomplete and uncertain knowledge of the network state. We give a MILP formulation of the problem that optimizes the cumulative time to restore the critical service demand and prove its NP-hardness, motivating the need for an efficient heuristic solution. We propose an algorithm called PRoTOn that guides recovery decisions while jointly deploying network monitors to facilitate the progressive acquisition of information on the network state. Through extensive simulations, we compare PRoTOn with several prior solutions to similar problems. Our experiments show that PRoTOn outperforms the existing approaches in all the considered network scenarios and in terms of all the performance indicators. It routes more service demands, earlier, with fewer repairs and with more efficient use of the monitor budget, thanks to the integration of tomographic techniques. We have provided public access to our code and data at [23].

References

[1] G. Aceto, A. Botta, P. Marchetta, V. Persico, and A. Pescapé, "A comprehensive survey on internet outages," *Journal of Network and Computer Applications*, vol. 113, pp. 36–63, 2018.

[2] The Outage Archive, https://puck.nether.net/pipermail/outages/, last accessed: July 28, 2022.

[3] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. E. Anderson, "Studying black holes in the internet with hubble." in *USENIX NSDI*, vol. 8, 2008, pp. 247–262.

[4] The Guardian, "Massive network outage in Canada hits homes, ATMs and 911 emergency lines," https://www.theguardian.com/world/2022/jul/08/internet-down-canada-rogers-mobile-network-outage, 2022, last accessed: July 11, 2022.

[5] R. Der Sarkissian, J.-M. Cariolet, Y. Diab, and M. Vuillet, "Investigating the importance of critical infrastructures' interdependencies during recovery; lessons from Hurricane Irma in Saint-Martin's island," *International Journal of Disaster Risk Reduction*, vol. 67, 2022.

[6] Cabinet Office of UK Government, "Ec-rrg resilience guidelines for providers of critical national telecommunications infrastructure," https://www.gov.uk/government/publications/ec-rrg-resilience-guidelines-for-providers-of-critical-national-telecommunications-infrastructure, 2021, last accessed: July 18, 2022.

[7] M. Bartock, J. Cichonski, M. Souppaya, M. Smith, G. Witte, and K. Scarfone, "Cybersecurity event recovery," *NIST Special Publication*, vol. 800, p. 184, 2016.

[8] L. Roberts and P. Pluto, "Communication during disaster recovery," https://openknowledge.worldbank.org/handle/10986/33685, 2020, last accessed: July 31, 2022.

[9] P. Fallara, "Disaster recovery planning," *IEEE Potentials*, vol. 23, no. 5, pp. 42–44, 2004.

[10] FEMA, "Puerto Rico Hurricane Maria," https://www.fema.gov/disaster/4339, last accessed: July 26, 2022.

[11] S. Ciavarella, N. Bartolini, H. Khamfroush, and T. La Porta, "Progressive damage assessment and network recovery after massive failures," in *IEEE INFOCOM*, 2017, pp. 1–9.

[12] J. Wang, C. Qiao, and H. Yu, "On progressive network recovery after a major disruption," in *IEEE INFOCOM*, 2011, pp. 1925–1933.

[13] D. Zad Tootaghaj, N. Bartolini, H. Khamfroush, and T. La Porta, "On progressive network recovery from massive failures under uncertainty," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 113–126, 2019.

[14] A. Bernstein, D. Bienstock, D. Hay, M. Uzunoglu, and G. Zussman, "Power grid vulnerability to geographically correlated failures—analysis and control implications," in *IEEE INFOCOM*, 2014, pp. 2634–2642.

[15] D. Z. Tootaghaj, N. Bartolini, H. Khamfroush, T. He, N. R. Chaudhuri, and T. La Porta, "Mitigation and recovery from cascading failures in interdependent networks under uncertainty," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 2, pp. 501–514, 2018.

[16] S. Ferdousi, M. Tornatore, F. Dikbiyik, C. U. Martel, S. Xu, Y. Hirota, Y. Awaji, and B. Mukherjee, "Joint progressive network and datacenter recovery after large-scale disasters," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1501–1514, 2020.

[17] G. Ishigaki, S. Devic, R. Gour, and J. P. Jue, "Deeppr: Progressive recovery for interdependent vnfs with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2386–2399, 2020.

[18] H. Yu and C. Yang, "Partial network recovery to maximize traffic demand," *IEEE Communications Letters*, vol. 15, no. 12, pp. 1388–1390, 2011.

[19] N. Bartolini, S. Ciavarella, T. F. La Porta, and S. Silvestri, "On critical service recovery after massive network failures," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2235–2249, 2017.

[20] N. Duffield, "Network tomography of binary network performance characteristics," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5373–5388, 2006.

[21] V. Arrigoni, N. Bartolini, A. Massini, and F. Trombetti, "Failure localization through progressive network tomography," in *IEEE INFOCOM*, 2021, pp. 1–10.

[22] L. Ma, T. He, A. Swami, D. Towsley, and K. K. Leung, "On optimal monitor placement for localizing node failures via network tomography," *Performance Evaluation*, vol. 91, pp. 16–37, 2015.

[23] NetMARS, https://github.com/matteoprata/Net-MARS.git.

[24] J. Qadir, A. Ali, K.-L. A. Yau, A. Sathiaseelan, and J. Crowcroft, "Exploiting the power of multiplicity: a holistic survey of network-layer multipath," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2176–2213, 2015.

[25] Cisco, "IP Routing," https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.html, last accessed: July 31, 2022.

[26] L. D. Valdez, L. Shekhtman, C. E. La Rocca, X. Zhang, S. V. Buldyrev, P. A. Trunfio, L. A. Braunstein, and S. Havlin, "Cascading failures in complex networks," *Journal of Complex Networks*, vol. 8, no. 2, 2020.

[27] L. Xing, "Cascading failures in internet of things: review and perspectives on reliability and resilience," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 44–64, 2020.

[28] M. Kim and J. S. Kim, "A model for cascading failures with the probability of failure described as a logistic function," *Scientific Reports*, vol. 12, no. 1, pp. 1–10, 2022.