Delay-Aware Routing in Software-Defined Networks via Network Tomography and Reinforcement Learning

Delay-Aware Routing in Software-Defined Networks via Network Tomography and Reinforcement Learning

Xu Tao, IEEE Student Member, Doriana Monaco, Alessio Sacco, IEEE Member, Simone Silvestri, IEEE Senior Member, and Guido Marchetto, IEEE Senior Member

***Abstract*—Numerous network management tasks in Software-defined networking (SDN) infrastructures, such as routing, resource allocation, and service placement, heavily depend on obtaining an accurate view of the network state. However, monitoring individual network elements incurs substantial overhead and often proves infeasible. To address this challenge, *network tomography* has emerged as a promising approach, capable of inferring the internal network state using end-to-end metrics observed by a limited set of nodes acting as monitors. Despite its potential, previous research in network tomography has not considered specific network management objectives and corresponding challenges, resulting in unsatisfactory performance. In this paper, we propose *Subito* (Shortest Path Routing with Multi-armed Bandits and Network Tomography), which integrates network tomography and *reinforcement learning* within software-defined networks to address the specific needs and challenges of delay-aware *shortest path routing*—a cornerstone of various network management tasks. By harnessing the capabilities of network tomography and reinforcement learning, Subito efficiently learns routing strategies with bounded regret, achieves minimal monitoring overhead, and maintains stable routing. Extensive experimental evaluations on synthetic networks and the GENI testbed show significant performance improvements of Subito versus two state-of-the-art approaches.**

***Index Terms*—Network Tomography, Reinforcement Learning, Multi-armed Bandits, Shortest Path Routing, Software-Defined Networking.**

I. Introduction

SDN has emerged as a transformative paradigm for quality of service (QoS)-aware network management, offering unprecedented flexibility and programmability. By decoupling the control plane from the data plane, SDN enables centralized network orchestration, dynamic traffic engineering, adaptive resource allocation, and network slicing for 5G [1], where ISPs can form a virtual network for each application [2]. A common trait of these network management tasks is the need for accurate knowledge of the current *network state* (e.g., delay, jitter, packet loss, and bandwidth) for both physical and virtual networks. However, it is widely recognized that acquiring such states through conventional IP-based network monitoring tools, such as the Simple Network Management Protocol (SNMP) and the Internet Control Message Protocol (ICMP), generates a high *monitoring overhead*. Additionally, this is often impossible due to the large scale, heterogeneity, multiple ownership, and partial observability of the networks. Therefore, it is essential to develop novel monitoring techniques that offer both low traffic overhead and complexity while ensuring accurate measurements, leveraging the flexibility of SDN.

Network Tomography [3]–[12] has recently received considerable attention as a lean approach for efficient monitoring, requiring only a small subset of network elements (i.e., nodes or links) to infer the internal QoS attributes while mitigating monitoring overhead. This technique relies on *end-to-end (e2e)* measurements collected by *monitors* generally placed at the edge of the network. The significant reduction in monitoring overhead compared to popular ICMP-based measurement tools and computational resources relative to packet-level and flow-level monitoring tools (e.g., NetFlow) endows network tomography with broad applicability across various domains, including the Internet [13], overlay networks [5], SDN-enabled 5G networks [1], and smart transportation [7]. Despite the significant efforts in this domain, current solutions have not adequately addressed the specific requirements of network management applications. Consequently, these solutions often overlook various challenges and fail to optimize their applicability within the context of the given application. Furthermore, numerous existing network tomography models [3]–[12] assume that monitors, located at the edge of the network, employ source routing to probe each other and obtain the e2e measurements. Nonetheless, this assumption proves impractical in real network infrastructures, primarily due to the lack of support for such features in standard off-the-shelf switches. To alleviate this assumption, network tomography can seamlessly combine with SDN and capitalize the SDN capabilities, such as controllable routing, programmable control and data plane, and a holistic view of the network state, for advanced network management schema [14]–[16].

In this study, we make advancements in the field of network tomography, with a specific focus on addressing the challenges of *shortest path routing*—an essential aspect of numerous network management tasks in all network architectures, including SDN-based networks. To achieve this objective, we introduce a novel approach named *Subito* (**S**hortest Path Ro**u**ting with Multi-armed **B**and**i**ts and Network **To**mography) which represents a pioneering integration of network tomography and

Xu Tao and Simone Silvestri are with the Department of Computer Science, University of Kentucky, Lexington, Kentucky, USA (e-mail: xu.tao@uky.edu; silvestri@cs.uky.edu)

Doriana Monaco, Alessio Sacco and Guido Marchetto are with DAUIN, Politecnico di Torino, 10129 Turin, Italy (e-mail: doriana.monaco@polito.it, alessio_sacco@polito.it; guido.marchetto@polito.it)

reinforcement learning. Unlike previous network tomography solutions, Subito stands out for its ability to swiftly learn routing paths even in the presence of noisy measurements. Moreover, by harnessing the customizable nature offered by the SDN paradigm, the proposed method ensures minimal monitoring overhead and maintains routing stability, all without the necessity of monitoring individual network elements.

Fig. 1 illustrates the overall framework of our proposed approach. Subito Routing Logic is in charge of selecting the shortest path with reduced monitoring overhead. And the SDN controller is employed to monitor the network, granting Subito access to a comprehensive network topology view and telemetry data. However, with Subito, we can mitigate the measurement overhead with network tomography and collect e2e metrics only from a subset of total switches, the *monitors*. We model the problem of learning the shortest path between a pair of monitors as a Multi-armed Bandits (MAB) problem, a subclass of *reinforcement learning*. At each time slot, an agent selects a path as the routing strategy to acquire the delays on that path, with the aim of balancing exploration and exploitation while facilitating knowledge exchange for faster learning. Notably, our formulation avoids action space explosion with network size, which sets it apart from existing solutions [17], [18]. Next, network tomography consolidates the set of selected paths into a linear system, and employing the theory of Matroids [19], it efficiently identifies a *basis*, i.e., a set of linearly independent vectors, to infer individual link metrics, i.e., link delays. However, due to frequent underdetermination of the system, certain links may have infinite solutions. To address this issue, we propose a method for estimating the delays of these *unidentifiable* links. Finally, the inferred or estimated individual link metrics are used to update the agents' collective knowledge and select the paths for the next time slot.

We prove that Subito has a bounded regret and conduct extensive experiments to assess its performance on both synthetic and real-world SDN networks. Specifically, we collected data on the National Science Foundation (NSF) GENI testbed [20] implementing an SDN controller. Results show substantial performance improvements of Subito versus two state-of-the-art approaches.

In summary, the main contributions of this paper are:

- We propose Subito, a novel approach that integrates network tomography and reinforcement learning, harnessing and complementing the inherent capabilities of SDN, to efficiently address the challenges of shortest-path routing in SDN-based architectures;
- We prove that *Subito* has a bounded regret;
- We conduct experiments in synthetic networks and two real networks, including the NSF GENI testbed. Results show up to three times reductions in monitoring overhead and regret (path delay) compared to two leading state-of-the-art solutions.

II. Related Work

In this section, we provide an overview of existing methodologies focused on inferring link-based metrics from e2e measurements. Additionally, we explore recent endeavors that leverage data-driven and Machine Learning (ML)-based techniques to facilitate routing packets. Particular emphasis is placed on approaches utilized for SDN scenarios.

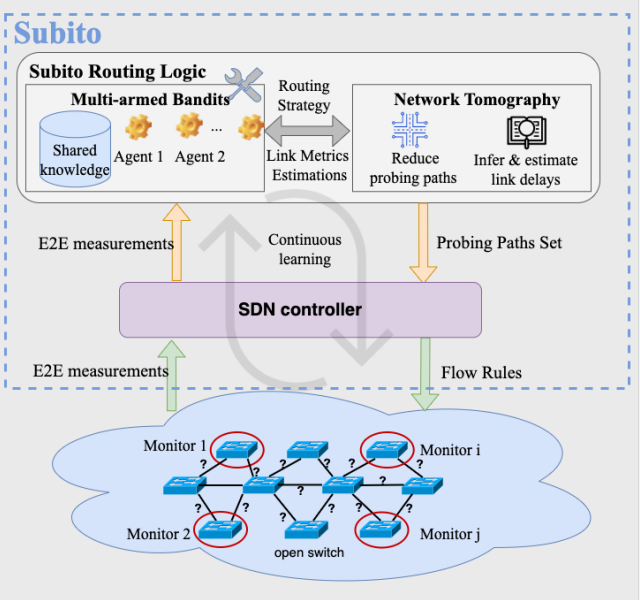


Fig. 1. Subito is built atop an SDN controller and exploits a minimal amount of network measurement to infer link metrics.

A. Network Tomography

Network tomography has attracted significant attention as it offers the capability to reduce the *monitoring overhead* in a network by inferring unknown internal link states solely based on end-to-end measurements [3]–[12], [21]. This technique primarily deals with *additive network metrics* commonly utilized in routing, such as link delay and packet loss. By selectively probing a limited set of paths from a small number of monitors positioned at the network's edge, network tomography constructs a linear system capturing the relationship between observed paths and corresponding link metrics. Numerous studies have explored the application of network tomography in various network scenarios. For instance, In [4], a solution is presented for inferring the network topology, while [5], [22] focuses on loss rate inference by selecting a minimal set of paths. Recently, [23] introduced the incorporation of path information to infer link metrics. Other research investigates network tomography's robustness under failures [6], [9], and explores security vulnerabilities of network tomography approaches [12]. A recent survey on network tomography is discussed in [11]. Additionally, recent approaches attempt to estimate additive metrics of slices, such as delays or logarithms of loss rates, using ML-based techniques, as seen in [24], where neural networks are employed.

One limitation of existing network tomography solutions is that the presence of noisy measurements in links' delays decreases the inference accuracy. Several studies have proposed approaches to address this issue, e.g., [25], [26]. For instance,

[3] proposes some monitor placement strategies to maximize link identifiability, aiming to accurately infer a larger number of links. Other work provides fundamental bounds [8], [10]. Nevertheless, despite these efforts, existing solutions for network tomography, including delay-based tomography, present several other limitations. First, most of these works rely on the assumption that monitors are equipped with the capability of source routing to probe each other to obtain the end-to-end delay. However, it is impractical in real network infrastructures, primarily due to the lack of support for such features in standard off-the-shelf switches. In this work, we alleviate this assumption by means of a seamless integration with an SDN controller, thus benefiting its controllable routing capability. Second, despite current extensive efforts in improving the inference performance of network tomography, there is a lack of research addressing the challenges of applying it in specific network management functions. In this work, we tackle the challenges of employing network tomography in the fundamental network function of efficient routing, pairing it with an advanced learning mechanism, i.e., reinforcement learning. Through an extensive evaluation, we demonstrate significantly better performance versus delay tomography.

B. Reactive Network Routing

Modern networks often experience unpredictable and variable traffic flows, resulting in link metrics, such as delay and packet loss, constantly varying. This variation naturally poses a challenge to determine efficient routing strategies since instantaneous measures may not reflect the nature of the actual link quality [27]–[29]. Several works have recently exploited ML, and in particular Reinforcement Learning (RL), to determine routing strategies in dynamic and noisy network environments. For example, the authors in [30] proposed CFR-RL, a schema designed to autonomously learn a policy for selecting critical flows based on the given traffic matrix. Subsequently, CFR-RL reroutes these identified critical flows to balance the link utilization of the network. The authors of [31] formulate the shortest path problem as a Markov decision process with tabular RL (Q-learning), where actions are determined in each decision iteration with hop-by-hop routing. The authors of [32] consider the shortest path learning problem in a stochastic wireless network environment under jamming attacks. Similarly, QR-SDN [33] demonstrates the utilization of Q-learning to minimize network latency by optimizing multipath routing in the context of an SDN architecture. In the same SDN context, RSIR [34] introduces a knowledge plane that works in conjunction with the management plane. The knowledge plane serves as a repository for data, which the SDN centralized controller then utilizes to determine the shortest routing path and distribute the load by using link-state information as the network state.

Similar to our work, the authors of [18], [27] formulate a reinforcement learning approach based on MAB and propose policies to learn the expected paths through exploitation and exploration strategies. While these works illustrate the benefit of ML, and specifically reinforcement learning, in determining routing strategies, they all assume the routing strategy to have *perfect and accurate* knowledge of the links' metrics. Gathering this information by directly monitoring each link not only results in significant monitoring overhead, but it is often impossible when internal network elements do not support or cooperate with the information-gathering protocols. While several recent studies attempt to alleviate measurement collection overhead yet utilizing Deep Reinforcement Learning (DRL) methods for routing, e.g., DRLS [35] and IQoR-LSE [36], none of them can statistically estimate link-based metrics. For example, IQoR-LSE [36] simplifies the DRL state by estimating a binary value (presence of congestion on a link) starting from a significant set of e2e QoS metrics.

In this study, we propose Subito, a novel approach that integrates network tomography and MAB techniques for the first time. Subito aims to expedite the learning of routing paths, ensure routing stability, and minimize monitoring overhead, complementing and enhancing the capabilities of SDN.

A preliminary version of this manuscript appeared as a short paper in [37]. The present version constitutes a substantial extension by: *i)* expanding the paper presentation with detailed explanations, additional figures, and deeper insights; *ii)* casting the problem in the domain of SDN; *iii)* providing a new algorithm to ensure minimum monitoring overhead, and the corresponding optimality proof; *iv)* expanding the experimental evaluation, which now includes the NSF GENI testbed, on a variety of performance metrics.

III. System Model and Problem Formulation

This section introduces the network model and problem formulation. Key notations employed in this paper are summarized in Table I.

TABLE I
SUMMARY OF KEY NOTATION

Symbol	Description
$\mathcal{G} = (V, L)$	Network $\mathcal{G}$, V the set of OpenFlow switches, L the set of physical or virtual links
$M \subseteq V$	Set of switches acting as monitors
$m_i \in M$	i-th monitor
$l_k \in L$	k-th link
X_k^t	Actual delay of link l_k at time t
$\hat{X}_k^t$	Inferred or estimated delay of link l_k at time t
d_k	Expected delay $\mathbb{E}(X_k)$ of l_k
$\mathbf{d}$	Set of expected delays of d_k, $k = 1, ..., \|L\|$
$\hat{d}_k^t$	Mean of observed delays of l_k at time t
$\hat{\mathbf{d}}$	Vector of mean observed delays $\hat{d}_k^t$, $k = 1, ..., \|L\|$
b_k^t	Number of observations of l_k at time t
$\mathbf{b}$	Vector of number of observations b_k^t at time t
p_{ij}^t	A path between monitors m_i and m_j
$p_{i,j}^*$	Shortest path between monitors m_i and m_j
P^t	Set of paths selected at time t
$\mathbf{P}^t$	Path matrix of P^t
$\mathbf{B}^t$	Basis of $\mathbf{P}^t$
B^t	Set of paths corresponding to $\mathbf{B}^t$
$y_{p_{ij}}^t$	End-to-end measurement of path p_{ij} at time t
$\mathbf{y}_{\mathbf{B}^t}^t$	Vector of end-to-end measurements of paths in $\mathbf{B}^t$

A. Network Model and Assumptions

Similar to prior literature on network tomography [4], [6], [8], [9], [12], we model the network topology as an undirected graph $\mathcal{G} = (V, L)$, where V and L represent the sets of OpenFlow switches and links (physical or virtual), respectively. Each link $l_k \in L$ is associated with an *additive metric*, such as delay [3], [23] or packet loss [5], as commonly assumed in network tomography. In this study, we focus on link delay since minimizing end-to-end latency is a prevalent objective for several applications, such as interactive and immersive applications [38]. We model the delay of each link as a random variable X_k, aligning with previous research in communication networks [39]. At time t, the delay of l_k is represented by the realization X_k^t of the random variable X_k. Following the experimental findings of [39], we assume that the distribution of X_k remains stationary over a sufficiently extended period, i.e., the type of the distribution and the expected value of X_k remain unchanged within this period. Note that, aside from stationarity of the series, Subito does not assume any specific distribution for these variables. However, we encounter two primary challenges: *(i)* the direct observation of X_k^t is unfeasible, and *(ii)* the expected value $d_k = \mathbb{E}(X_k)$ remains unknown. The collection of all expected values is denoted as $\mathbf{d} = d_k, k = 1, \ldots, |L|$. Furthermore, we impose no restrictions on the type or dependency of the distribution of delays across links.

A subset of access switches denoted as $M \subseteq V$ serve as *monitors* in the network. These monitor switches are typically located at the network's edge, providing external access to the network for customers or other connected networks. We assume the locations of the monitors are predefined in the network. However, numerous research endeavors [3], [4], [11] have concentrated on refining monitor deployment strategies to enhance the inference accuracy of network tomography. These strategies are orthogonal to Subito and can be easily integrated with it. Centralized control of the network is achieved through the use of a single SDN controller. In our SDN scenario, the controller only communicates with the *monitors* to collect *end-to-end* delays between them by exchanging probing packets. Here, we assume that the communication between the SDN controller and monitors is realized through southbound channels, not shared with the data links. We provide a discussion about the advantages of our approach when such assumption does not hold, and consequently southbound channels are shared with data links, in Section VII. We adopt the assumption as in prior research [3]–[5], [9], that link delays are *additive*. This implies that when a path p_{ij}^t is probed between two monitors $m_i, m_j \in M$ at time t, the resulting end-to-end measurement is given by $y_{p_{ij}}^t = \sum_{l_k \in p_{ij}^t} X_k^t$. These measurements serve as the foundation for our network tomography inference framework. Once the SDN controller collects these *end-to-end measurements*, our network tomography module can utilize them to infer the link delays. Then, Subito can make delay-aware routing decisions based on this information by utilizing an MAB-based learning technique, as described in Section IV-A.

B. Problem Statement

The primary objective of this research is to efficiently learn the optimal routing, represented by the set of shortest paths, among the monitors M within an unknown network environment. These shortest paths are determined with respect to the expected delays $\mathbf{d}$. The environment remains unknown due to three main factors: *i)* direct observation of link delays is not feasible, *ii)* link delays are intrinsically dynamic, following the distribution of variables X_k for $l_k \in L$, and *iii)* the expected delays $\mathbf{d}$ are not known. In addition to being effective, we aim for the learning process to be highly *efficient*, causing minimal monitoring overhead, and to provide *routing stability*, i.e., converging towards the optimal path and avoiding continuous oscillations across multiple paths that would cause performance instability and poor resource utilization.

IV. Subito

As shown in Fig. 1, Subito has three main components: *(i)* an SDN controller to have a centralized view of the network topology and interact with data plane, *(ii)* a MAB algorithm for shortest path selection, *(iii)* network tomography to reduce monitoring overhead. The MAB and network tomography components constitute the Subito Routing Logic. We depict in Fig. 2 the information flow of our system. The MAB algorithm is constituted by multiple agents, where each agent is responsible for selecting the shortest path of its responsible monitor pair. The SDN controller probes the reduced path set by collecting e2e measurements from the network environment (with unknown link delays) and installs flow rules associated with the selected paths. Subsequently, network tomography utilizes these e2e measurements to estimate the link delays, which are then fed back to the MAB component to update the knowledge of link delays and make decisions for the next iteration.

We outline these steps in Algorithm 1, and we elaborate on each of these main components in the rest of this section.

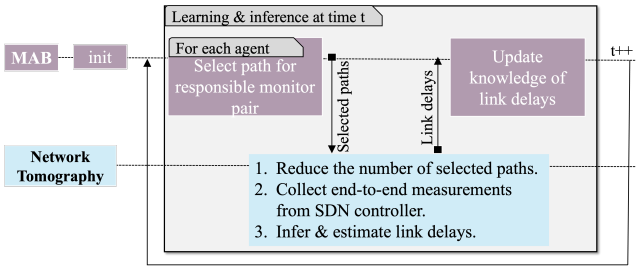


Fig. 2. Subito Information Flow

A. Learning Routing through Multi-Armed Bandit

MAB is a *reinforcement learning* technique where an agent aims at learning the best policy to optimize an objective [27], [32]. In our scenario, an agent is assigned to each pair of monitors m_i and m_j from the set M. At every decision slot t, each agent needs to employ a policy to choose an *action*, which involves selecting a path p_{ij}^t between m_i and m_j. Conversely, the non-monitor switches are not involved in the decision

process, but traffic might flow over them. In this case, they just forward packets using the routes selected by agents.

We denote the action space for each agent as $\mathcal{F}_{ij}$ (i.e., the possible paths between m_i and m_j) and the set of paths selected at time t by all agents as $P^t \subseteq \bigcup_{m_i,m_j \in M} \mathcal{F}_{ij}$. It can be noted that, although the action space $\mathcal{F}_{ij}$ grows exponentially with the size of the network, Subito allows agents to efficiently navigate this expansive action space and prevents exploring every path within it, as detailed below.

When actions are played by agents, end-to-end measurements are observed. By exploiting network tomography (detailed in Section IV-B), we are able to infer and estimate individual link delays from end-to-end measurements with a minimum monitoring overhead. Since such links are shared among paths between different agents, Subito allows agents to share the gathered knowledge, thus achieving faster learning. The *state* is given by this collective knowledge of individual links observed by agents through two vectors, $\hat{\mathbf{d}}$ and $\mathbf{b}$. These vectors store the mean of observed delay samples for each link l_k, and the number of times l_k has been observed, respectively. This state, used to determine the best action at each iteration, is shared among all agents to enhance the scalability of the learning process and to allow agents to efficiently navigate the action space. The *reward* for each agent associated with the monitor pair m_i, m_j at time slot t is the observed end-to-end delay of the selected path p_{ij}^t, denoted as $y_{p_{ij}}^t = \sum_{l_k \in p_{ij}^t} X_k^t$.

Let p_{ij}^* represent the optimal *shortest path* between m_i, m_j, and $y_{p_{ij}^*} = \sum_{l_k \in p_{ij}^*} d_k$ be the corresponding *expected reward*. We recall that d_k is the expected value of the variable X_k. We define the *regret* $\Re^t$ at time t, as the sum of the differences between the cumulative reward obtained by the paths selected by MAB and the expected reward achieved by always selecting the optimal shortest paths for all agents. Formally, it is represented as:

$$\Re^t = \sum_{m_i,m_j \in M} \sum_{t=1}^{T} \left(y_{p_{ij}}^t - y_{p_{ij}^*} \right) \tag{1}$$

Note that, agents only use reward for their decisions. The regret is only considered to assess the strategy learned by agents in Theorem 3.

As shown in Algorithm 1, Subito is composed of two phases: initialization phase, and learning and inference phase.

1) Initialization Phase: The initialization phase (lines 1-7) aims to ensure that we gather a minimum amount of knowledge for each random variable, i.e., link delay. For this purpose, a for loop iterates over the set of links. At the k-th iteration, we guarantee that link l_k will be probed. This is achieved by selecting a pair of monitors (m_i, m_j) and finding a simple path from m_i to an end point of l_k, and another simple path from the other end point to m_j. These paths can be found, for example, using Dijkstra's algorithm [40]. The path p^t resulting from the concatenation of these paths with l_k is subsequently probed by the SDN controller to collect end-to-end measurements y_p^t. However, since we cannot observe the individual delay of the links in p^t, and network tomography is ineffective when a single path is probed, we estimate the delay of each link in p^t using the average end-to-end delay $\frac{y_p^t}{|p^t|}$. This information is stored in the vectors $\hat{\mathbf{d}}$ and $\mathbf{b}$, which are updated for all the links in p^t, including l_k, as follows:

$$\hat{d}_r^t = \begin{cases} \frac{\hat{d}_r^{t-1} b_r^{t-1} + \frac{y_p^t}{|p^t|}}{b_r^{t-1}+1} & \text{if } l_r \in p^t \\ \hat{d}_r^{t-1} & \text{otherwise} \end{cases} \quad b_r^t = \begin{cases} b_r^{t-1}+1 & \text{if } l_r \in p^t \\ b_r^{t-1} & \text{otherwise} \end{cases} \tag{2}$$

Equation (2) , given a link l_r, updates $\hat{d}_r^t$ if l_r belongs to the current probed path p^t. Specifically, given the new estimated delay $\frac{y_p^t}{|p^t|}$, it updates the previous average $\hat{d}_r^t$ with this new value, considering the number of samples b_r^t used to calculate that average. Additionally, it increases b_r^t by one. If l_r is not in p^t, $\hat{d}_r^t$ and b_r^t are left unchanged.

2) Learning and Inference Phase: During this phase (lines 8-23), at each time slot t, a path p_{ij}^t is selected by an agent for each pair of monitors m_i and $m_j \in M$ according to the following Equation (3):

$$p_{ij}^t = \underset{p \in \mathcal{F}_{ij}}{\arg\min} \sum_{l_k \in p} \left(\hat{d}_k^{t-1} - \sqrt{\frac{(|L|+1)\ln t}{b_k^{t-1}}} \right) \tag{3}$$

Algorithm 1: Subito

```
/* Initialization phase                                              */
1  t = 0; d̂ = [0]_{1×|L|}; b = [0]_{1×|L|};
2  for k=1 to |L| do
3      t = t + 1;
4      Probe a path p^t s.t. l_k ∈ p^t and collect the end-to-end
         measurement y_p^t;
5      for each l_r ∈ L do
6          Update d̂_r^t = { (d̂_r^{t-1} b_r^{t-1} + y_p^t/|p^t|) / (b_r^{t-1}+1)   if l_r ∈ p^t
                          { d̂_r^{t-1}                                            otherwise
7          Update b_r^t = { b_r^{t-1} + 1   if l_r ∈ p^t
                          { b_r^{t-1}       otherwise
/* Learning and Inference phase                                      */
8  while True do
9      t = t + 1;
10     P^t = {};
11     for any two monitors m_i, m_j ∈ M do
12         p_ij^t = argmin_{p ∈ F_ij} Σ_{l_k ∈ p} (d̂_k^{t-1} − sqrt((|L|+1) ln t / b_k^{t-1}));
13         P^t = P^t + p_ij^t;
       /* Probing and Inference with Network
          Tomography                                                 */
14     Build binary path matrix P^t from P^t;
15     Find the optimal basis B^t of P^t with Algo. 2, s.t. B^t ⊆ P^t;
16     Probe paths in B^t with SDN controller, observe end-to-end
         measurements y_{B^t}^t;
17     Build linear system B^t x^t = y_{B^t}^t;
18     Infer and estimate the delay of links in L_prob^t by solving the
         linear system;
19     for each l_k ∈ L_prob^t do
20         X̂_k^t = { X_k^t                                              if l_k ∈ ⋃_{q∈B^t} L_q^id
                   { min_{p_q ∈ B^t s.t. l_k ∈ p_q} (y_q^t − Σ_{l_k ∈ L_q^id} X_k^t) / |L_q^u|   otherwise
21     for each l_k ∈ L do
22         Update d̂_k^t = { (d̂_k^{t-1} b_k^{t-1} + X̂_k^t) / (b_k^{t-1}+1)   if l_k ∈ L_prob^t
                          { d̂_k^{t-1}                                       otherwise
23         Update b_k^t = { b_k^{t-1} + 1   if l_k ∈ L_prob^t
                          { b_k^{t-1}       otherwise
```

Intuitively, the agent selects the path based on the current knowledge of each link delay stored in $\hat{\mathbf{d}}$. However, the delay $\hat{d}_k^{t-1}$ of link l_k is modified by an amount of - $\sqrt{\frac{(|L|+1)\ln t}{b_k^{t-1}}}$, where L denotes the set of links, t the current time slot, and b_k^{t-1} the number of times link l_k has been probed. The adjustment of $\sqrt{\frac{(|L|+1)\ln t}{b_k^{t-1}}}$ is derived using Chernoff-Hoeffding inequality [41] and it is established as an upper bound ensuring that the deviation from the expected delay of each link is within this bound. Moreover, this term plays a pivotal role during the exploration phase by facilitating the selection of links with fewer measures, as numerous observations allow for a more accurate determination of the true expected delay. Over time, the number of observations increases (b_k^{t-1} becoming larger), leading the adjustment term $-\sqrt{\frac{(|L|+1)\ln t}{b_k^{t-1}}}$ towards 0. Ultimately, the algorithm transitions from exploration to exploitation and converges to the optimal paths with the learned knowledge during exploration, ensuring the stability of the routing strategy. We refer the reader to [41] for more details.

We store all selected paths in a set P^t. This set is initially processed using *network tomography* to find a subset $B^t \subseteq P^t$ that possesses the same inference power as P^t but requires minimum monitoring overhead. The calculation of B^t is explained in Section IV-B. Then, the paths within the basis B^t are probed by SDN controller, and the resulting end-to-end measurements are observed. Let L_{prob}^t represent the set of links of the paths in B^t. Subito utilizes network tomography to infer and estimate the individual delay of the links in L_{prob}^t. Specifically, for each link $l_k \in L_{prob}^t$, we calculate a new realization $\hat{X}_k^t$ of the random variable X_k. It is important to note that, as detailed in Section IV-B, the estimation $\hat{X}_k^t$ may slightly differ from the actual realization X_k^t, especially for unidentifiable links. Finally, $\hat{X}_k^t$ is used to update the vectors $\hat{\mathbf{d}}$ and $\mathbf{b}$.

B. Probing and Inference with Network Tomography

In the following, we outline how Subito utilizes network tomography to minimize network monitoring overhead and estimate individual link delays based on end-to-end measurements.

1) Reduce Monitoring Overhead: Let $\mathbf{P}^t$ be a binary path matrix of size $|P^t| \times |L|$, representing the links in the set of paths P^t selected by Subito at time slot t. Specifically, for each $p_q \in P^t$, $\mathbf{P}_{q,k}^t = 1$ if path p_q contains link l_k, and $\mathbf{P}_{q,k}^t = 0$ otherwise. In addition, we denote the observed end-to-end path measurements in P^t by the column vector $\mathbf{y}^t = (y_1^t, ..., y_q^t)^T$. Due to the additive nature of delay [42], it is possible to formulate a linear system as follows:

$$\mathbf{P}^t \mathbf{x}^t = \mathbf{y}^t \tag{4}$$

where $\mathbf{x}^t = (X_1^t, ..., X_{|L|}^t)^T$ represents a column vector containing the delay realizations of all links at time t.

As in our network tomography model, we reduce the *monitoring overhead* by solving a linear system, we recall that the concept of *basis* in such a linear system, referred to as $\mathbf{B}^t$, represents the maximal set of linearly independent paths, with each path considered as its vector representation in $\mathbf{P}^t$. It has been shown that a basis provides the same inference power as $\mathbf{P}^t$ [7]. Specifically, given $\mathbf{y}_{\mathbf{B}^t}^t$ as the column vector containing only the end-to-end measurement of paths in $\mathbf{B}^t$, the following linear system induced by $\mathbf{B}^t$:

$$\mathbf{B}^t \mathbf{x}^t = \mathbf{y}_{\mathbf{B}^t}^t \tag{5}$$

has the same solutions as the system in Eq. 4 [7].

However, not all bases are equal in terms of the resulting monitoring overhead. While all bases have the same cardinality (number of paths), bases may differ in terms of the overhead induced by monitoring such paths. Here, we assume that the overhead is proportional to the number of probe packets introduced in the network, and thus to the length (number of hops) of a path. Consequently, let B^t be the set of paths in a basis $\mathbf{B}^t$, we define the cost $C(B^t) = \sum_{p_q \in B^t} |p_q|$, where $|p_q|$ is the length of p_q. Our objective is to find a basis of $\mathbf{P}^t$ with minimum overhead, that is:

$$\begin{aligned} \min_{B^t \subseteq P^t} \quad & C(B^t) \\ \text{s.t.} \quad & \mathbf{B}^t \textit{ is a basis of } \mathbf{P}^t \end{aligned} \tag{6}$$

We can optimally solve the above problem using the theory of Matroids [19]. Specifically, we can design a greedy algorithm as shown in Alg. 2. The algorithm takes the set of paths P^t as input. It initializes an empty solution set B^t (line 2), then sort paths in P^t by increasing overhead (i.e., increasing length) (line 3). At each iteration of the for loop (lines 4-7), the path p_i is considered, according to the sorted order. If the rank increases by adding p_i to B^t , i.e., if p_i is linearly independent with respect to the paths in B^t, then p_i is added to B^t. The rank can be calculated using Gaussian Elimination (GE) [42]. Finally, path matrix $\mathbf{B}^t$ is returned (line 8).

Algorithm 2: Find minimum basis

```
1 Input: Path set P^t;
2 B^t = ∅ ;
3 Sort paths in P^t by increasing overhead;
4 for i = 1, ..., |P^t| do
5     Let B^t_{p_i} be the path matrix of B^t ∪ p_i;
6     if rank(B^t) < rank(B^t_{p_i}) then
7         B^t = B^t + p_i;
8 return B^t;
```

We can show that Algorithm 2 finds the optimal basis using the theory of Matroids.

Definition 1 (Matroid [19])**.** *A matroid $\mathbb{M}$ is a pair (U, S), where U is a finite ground set and S is a non-empty collection such that $\mathsf{S} \subseteq 2^{\mathsf{U}}$, with the following properties:*

- *$\forall A \subset D \subseteq U$, if $D \in \mathsf{S}$, then $A \in \mathsf{S}$.*
- *$\forall A, D \in \mathsf{S}$ with $|D| > |A|$, $\exists x \in D \setminus A$ s.t. $A \cup \{x\} \in \mathsf{S}$*

The following theorem proves the optimality of Algorithm 2.

Theorem 2. *([19]). Let $\mathbb{M} = (\mathsf{U}, \mathsf{S})$ be a matroid and $C : 2^U \to \mathbb{R}_+$ a modular function, i.e., $C(A) = \sum_{a \in A} C(\{a\})$, $\forall A \in \mathsf{S}$. Then, the greedy algorithm finds an optimal solution for minimizing $C(S)$ subject to $S \in \mathsf{S}$.*

Proof. It is well-known that the notion of linear independence over a set of vectors forms a matroid [6]. Additionally, the function $C()$ is modular since it sums the number of links in a path independently. As a result, the algorithm is optimal. □

2) Link Delay Inference with End-to-End Measurements: After probing the paths in B^t, we obtain the vector of end-to-end measurements $\mathbf{y}^t_{\mathbf{B}^t}$. We can then formulate and solve the linear system[1] in Eq. (5). Unfortunately, this linear system is often undetermined. Consequently, a unique solution exists only for a subset of links [3], [8], referred to as *identifiable* links, while the remaining links are *unidentifiable*, i.e., they have an infinite number of possible solutions. With Subito, we provide an estimation for both *identifiable* and *unidentifiable* links.

Given a path $p_q \in \mathbf{B}^t$, L^{id}_q the set of identifiable links, and L^u_q the set of unidentifiable links in p_q, for each link l_k in p_q, if $l_k \in L^{id}_q$, we can obtain its unique solution X^t_k. However, if $l_k \in L^u_q$, we calculate an estimation $\hat{X}^t_k$ as follows.

We start by calculating the amount of delay attributed to unidentifiable links in path p_q at time t, denoted as $y^t_q - \sum_{l_k \in L^{id}_q} X^t_k$. We then average this amount over all unidentifiable links in L^u_q. This result serves as an estimate of the delay of the links in L^u_q. However, it is important to note that an unidentifiable link may appear in multiple paths and each of these paths provides an estimate for that link. Therefore, we assign an approximate delay to an unidentifiable link by considering the minimum of the estimates provided by the paths it appears in. More formally, the estimate $\hat{X}^t_k$ for a link $l_k \in L^u_q$ is given by:

$$\hat{X}^t_k = \min_{p_q \in \mathbf{B}^t \text{ s.t. } l_k \in p_q} \frac{y^t_q - \sum_{l_k \in L^{id}_q} X^t_k}{|L^u_q|} \tag{7}$$

Such estimated delay is utilized by Subito to update $\hat{\mathbf{d}}^t$ and $\mathbf{b}^t$ for the subsequent iterations, as shown in lines 22-23 of Algorithm 1. The approximation approach with the minimum estimation for unidentifiable links is simple, efficient, and effective. Although this approach might cause discrepancies between estimated and actual delays at certain time slots, this effect can be mitigated during the learning process. As Subito encourages exploration at the beginning of the learning phase, the estimated link delay $\hat{X}^t_k$ is determined by different sets of probing paths at different iterations. And the mean value of link delay, stored in $\hat{\mathbf{d}}$, is updated by averaging over the accumulative historical estimations of $\hat{X}^t_k$. As the observation frequency of each link increases during the exploration phase, $\hat{\mathbf{d}}$ becomes more accurate. The experimental results in section V demonstrate the accuracy and efficiency of this approach.

C. Bounded Regret

We also demonstrate how the regret provided by Subito for identifiable links is bounded.

[1]Note that, monitors exchange multiple probing packets when calculating the end-to-end delay of a path. This makes it unlikely that all packets will be lost or dropped while probing. Nevertheless, in the unlikely event of this happening, the SDN controller loses visibility of that path and consequently the path remains absent from the linear system in the current iteration.

Theorem 3. *Provided that all links are identifiable, Subito has a bounded regret given by:*

$$\Re^t \leq |M|^2 \left[\frac{4|L|^3(|L|+1)\ln t}{(\Delta_{min})^2} + |L| + \frac{\pi^2}{3}|L|^2 \right] \Delta_{max}, \tag{8}$$

where Δ_{min} and Δ_{max} represent the minimum and maximum regrets of actions, while $|M|$ and $|L|$ refer to the number of monitors and links, respectively.

Proof Sketch. Our approach extends the solution presented in [41] by incorporating multiple agents with shared knowledge. In particular, our formulation involves an agent for each pair of monitors in M, and the observed realizations are shared through the vectors $\hat{\mathbf{d}}^t$ and $\mathbf{b}^t$. As a result, agents can collectively gather more information at each time slot. Therefore, the theorem follows based on the regret bound provided in [41] and our definition of regret in Eq. 1. □

V. Experimental Evaluation

In this section, we evaluate Subito versus two state-of-the-art approaches using synthetic and realistic networks. The first comparison approach exploits MAB, while the second one employs network tomography. Before describing our results in which Subito outperforms these two techniques, we first describe the comparison approaches and then the experimental setup.

A. Benchmarks Description

The first benchmark, called *UCB1* [18], is based on a MAB approach where each agent (monitor pair) learns independently. Differently from Subito, the action space for UCB1 are all possible paths between a monitor pair, and paths' delays are learned individually. Thus, UCB1 does not rely on network tomography, since it does not work at the granularity of links. However, this approach prevents agents from sharing information. It is essential to note that the state space for each agent in UCB1 is defined by the number of paths between the corresponding monitor pair, and thus it grows exponentially with the network's size. For a fair comparison, we limit the size of the state space for each monitor pair to $2^{\frac{|V|}{10}}$, where V represents the number of nodes. This constraint is a significant lower bound compared to the total number of paths. To populate the state space, we randomly include paths, ensuring the shortest path is among the available paths (refer to [18] for more details). By comparing our approach with UCB1, we can evaluate the benefit of network tomography, sharing knowledge across agents, and having a finer grain modeling of the action space at the link level.

The second benchmark is a recently proposed network tomography approach, named *Bound-based Network Tomography (BoundNT)* [21]. This approach utilizes end-to-end measurements to decrease the overhead and infer internal link delays using network tomography. Additionally, BoundNT calculates an upper bound b_{min} on the delay of each unidentifiable link l_k at each time slot t. Such bound is used to estimate the realization of the random variable associated to

each unidentifiable link l_k, i.e., $\hat{X}_k^t = b_{min}$. Unlike Subito, however, BoundNT is not designed for a specific network management task. In order to adopt this method for routing, since BoundNT does not learn over time, we calculate the shortest paths between monitor pairs at round t based on the instantaneous delays inferred at round $t-1$. BoundNT employs network tomography to provide inference capability with low monitoring overhead. However, it lacks learning over time. By comparing Subito with BoundNT, we can evaluate the importance of including a learning process in determining optimal shortest paths to achieve better routing behavior.

Finally, to evaluate the efficiency of the proposed delay estimation method for unidentifiable links, in Eq. (7), we compare Subito to *Subito**. Subito* is a version of Subito with zero estimation error that is able to observe the individual link delays of each probed path. Table II summarizes the main features of Subito, UCB1, and BoundNT.

TABLE II
MERITS OF SUBITO, UCB1, AND BOUNDNT

Method	Learning	Reduced Overhead
Subito	✓	✓
UCB1	✓	
BoundNT		✓

B. Experimental Networks

We evaluate Subito, Subito*, UCB1 and BoundNT on both synthetic and realistic networks.

Synthetic networks. Network typologies are generated with the Barabasi-Albert (BA) model [43]. Nodes are added iteratively, each new node attaches to 2 existing nodes using the preferential attachment. This is well-known to result in Internet-like scale-free networks. We then assign the link delays with realistic distributions. The variable X_k for link l_k can be modeled as an exponential distribution with a parameter λ_k, randomly chosen in the interval $[0, 1]$, as discussed in [39]. As a result, the unknown *expected delay* of l_k is $d_k = \frac{1}{\lambda_k}$.

Real networks. We first consider the real-world topology BTN from the Internet Topology Zoo [44]. We used synthetic delays in this case, as for BA networks. In order to consider a more realistic case, we extended an SDN controller over the GENI testbed using an NSF-like [45] network. We collected link delay measurements by utilizing a triangulation method as in [46]. Packets marked with a specific IP protocol number, e.g., 253, notify the switch to insert the current timestamp in the IP Option field before forwarding the packet. Once this type of packet traverses the initial switch, it is directed to the adjacent switch and subsequently forwarded to the SDN controller. Upon receiving the packet via the Packet-In message, the controller compares the current system time with the timestamp in order to determine the loop delay. This delay encompasses both the link delay between the two switches and the delay of the control link. To ascertain the control link delay, the module simultaneously sends an Echo message to each switch, utilizing the symmetric message to obtain the necessary information. By subtracting the control link delay from the loop delay, we can calculate the delay of a specific link. This process is repeated at every link, while the end-to-end delay is also computed easily from the final destination. We collect the data for a total of 12 hours where delay information is computed every 10 seconds. Although the NSF network is of relatively small size, it allows us to consider a real-world scenario where delays are not artificially generated but collected based on real traffic. The size of this testbed is limited by physical limitations and resource constraints inherent of the experimental infrastructure.

In all experiments, we randomly select switches as monitors. Since our goal is to learn the shortest paths between monitors, we prune the network from all links that are not included in any simple path between any pair of monitors. These links, by definition, do not belong to any shortest path. In fact, in order for these to belong to a path between two monitors, the path must contain a cycle. An example of these links (in green and thick) is shown in Figure 3. To simplify the experiments and concentrate on the goal, these links are omitted from the network topology in the experiments.

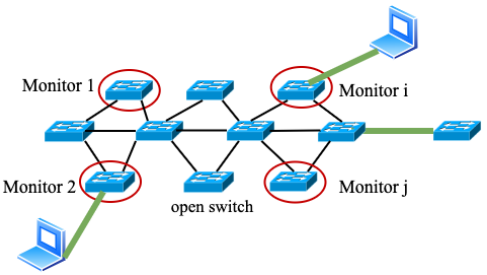


Fig. 3. An example of links (in green and thick) that are not included in any simple path between any pair of monitors.

Table III summarizes the adopted topologies, including network type, network size, and the path space for UCB1. We do not impose any bound on the path space which composes the action space of Subito. The average degree for BTN and NSF is 2.5. Table IV illustrates the statistical summary for both the synthetic dataset and dataset collected from GENI. The parameters μ and σ^2 correspond to the average link delays and delay variances, respectively, across all links in the datasets.

TABLE III
NETWORK TOPOLOGY USED IN EXPERIMENTS

Network Type	#nodes	#links	#paths space for UCB1
BA	20	36	4
BA	40	76	16
BA	50	96	32
BA	60	116	64
BA	80	156	256
BTN	53	65	32
NSF	11	14	2

C. Experiment Results in Synthetic Networks

We run Subito, Subito*, UCB1 and BoundNT over 3000 time slots over multiple runs in BA networks and BTN net-

TABLE IV
STATISTICS OF LINK DELAY DATASETS

Dataset	μ (ms)	σ^2 (ms)	samples of each link
synthetic	(0,10]	(0, 85]	4000
GENI testbed	(27, 66]	(0, 209]	4320

work with synthetic link delay distribution. In the experiments, we show the resulting average and standard deviation.

Experiment I: Scalability over the percentage of monitors in synthetic BA network. In the first experiment, we vary the percentage of switches acting as monitors from 10% to 50% in the synthetic network with 50 nodes. We first study the *learning error* overtime incurred by Subito during the learning phase. The error at time t is calculated using the Mean Square Error (MSE) of the learned averages of the random variables $X_1, \ldots, X_{|L|}$ stored in the vector $\hat{\mathbf{d}}^t$ with respect to the actual averages in $\mathbf{d}$.

Fig. 4 shows the results. Initially, the error is high irrespective of the number of monitors available. This is due to the inaccurate knowledge gathered during the initialization phase, where single paths are probed and network tomography is not adopted. Nevertheless, as soon as the learning phase starts, Subito is able to quickly learn the expected link delays and reduce the learning error. There are two main points to note in Fig. 4. First, as we increase the number of monitors, the error significantly reduces due to the increased ability of network tomography in estimating the individual link delays as more paths are probed between monitors. Second, the error stabilizes and never reaches zero. Since our goal is to discover the shortest paths between monitors, we do not need to perfectly learn the expected delays of all links. As Subito gathers knowledge through exploration, it gradually switches to exploitation to use the shortest path that has been learned.

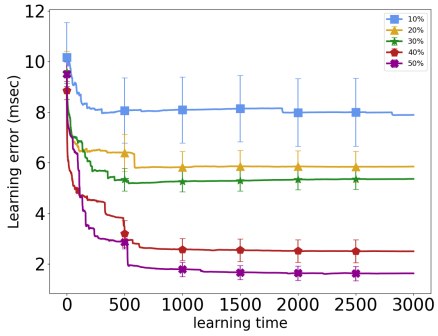
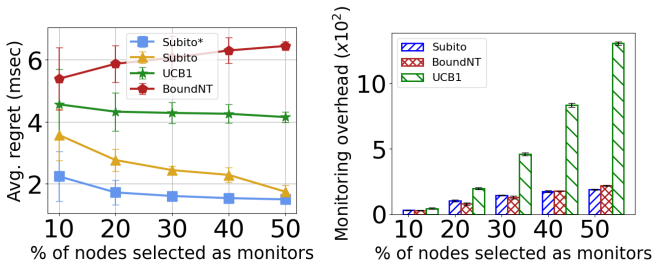


Fig. 4. Experiment I: MSE over time of learned delays under different % of monitors using BA network with 50 nodes.

Next, we study the *average regret*. The average regret can be interpreted as the average additional delay incurred by not selecting the shortest paths. It is calculated as the regret defined in Eq. 1 divided by the total time and the number of monitor pairs. This normalization is necessary to compare regret across different scenarios. The results are presented in Fig. 5(a). BoundNT suffers from the lack of learning, resulting in an increasing regret as more monitors become available. Subito*, Subito, and UCB1 exhibit decreasing regret due to the increased information gathered among agents. Subito* achieves the best performance, since it can observe the actual real-time delay realization of each individual link. Note that, although this eliminates the estimation error attributed to the unidentifiable links introduced by network tomography, it still incurs in a regret larger than zero. This is due to the need of learning the links' expected delays, and thus the optimal paths. However, the gap between Subito* and Subito reduces as more monitors become available, as network tomography becomes more accurate. On the other hand, UCB1 shows poor performance due to its inability to share knowledge and efficiently explore the state space. As a numerical example, UCB1 and BoundNT show a regret that is up to twice and three times higher than Subito, respectively.

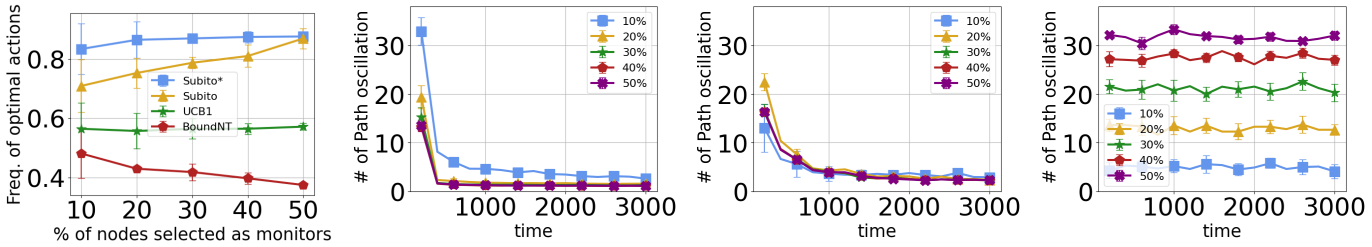


(a) (b)

Fig. 5. Experiment I: Average regret (a) and Monitoring overhead (b) varying the % of monitors using BA network with 50 nodes.

We now consider the *monitoring overhead* introduced by probing paths between monitors, as defined in Section IV-B1. At each iteration, we calculate the overhead and average it over the entire experiment. Results are shown in Fig. 5(b). Subito and BoundNT show similar performance thanks to the ability of *network tomography* to reduce the monitoring overhead. It is worth noting that, in some cases, BoundNT has a slight less overhead than Subito, which appears to contradict Theorem 2. It is important to realize that these two approaches work on a different set of paths. BoundNT selects a random basis of all paths between monitors, while the MAB phase of Subito returns the set of paths to be probed balancing exploration and exploitation. It may occur that BoundNT achieves less overhead. Nevertheless, the overhead of Subito is overall within 5% of BoundNT. Although BoundNT offers monitoring overhead comparable to Subito, attributed to the benefits of network tomography, it faces a significantly higher regret, and unstable routing behaviors, as shown in the next set of experiments, due to the absence of a learning mechanism. Conversely, the monitoring overhead incurred by UCB1 increases tremendously as the number of monitors increases, reaching about six times that of Subito and BoundNT. This is due to agents monitoring paths independently, paired with the increase of the state space under UCB1. Overall, the results in Fig. 5 (a) and (b) demonstrate the benefits of combining *network tomography* with MAB in achieving low regret as well as low monitoring overhead.

The next set of experiments focuses on the *quality* of routing. Specifically, we examine the *frequency optimal actions*,



(a) (b) (c) (d)

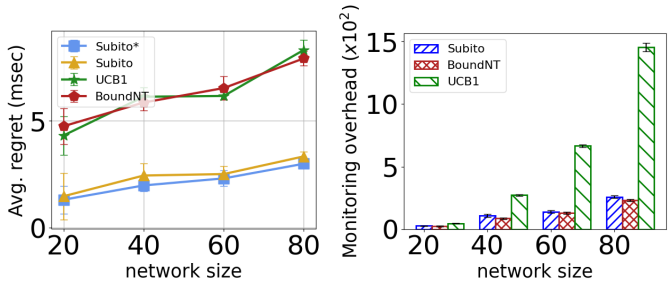
Fig. 6. Experiment I: Freq. of optimal actions (a) and path oscillations under Subito (b), UCB1 (c), and BoundNT (d) under different % of monitors in BA network with 50 nodes.

which refers to how often the actual shortest path is selected, and *path oscillations*, which measures the stability of the routing. Fig. 6(a) shows the frequency of optimal actions for Subito, Subito*, BoundNT, and UCB1 under different percentages of monitors. Here we focus on the last 1000 time slots out of the 3000 considered. Although the delay bounds for unidentifiable links becoming more accurate as the percentage of monitors increases under BoundNT, this approach exhibits a decreased rate of optimal actions. This is due to the increasing difficulty of selecting the shortest paths using only the latest delays when more potential paths are available. Similarly, UCB1 does not scale well with the number of available monitors. In contrast to Subito, the agents in UCB1 learn paths independently and cannot share the collected knowledge. Each agent learns within its own action space, so increasing the number of available monitors does not improve performance. Conversely, Subito significantly outperforms UCB1 and BoundNT. The frequency of optimal actions improves as more monitors are deployed since more monitors allow more paths to be probed. This allows agents to collect and share more information, and it improves the ability of network tomography to provide accurate inferences. Noticeably, Subito achieves performance close to Subito*, showing the inferencing power of network tomography and the proposed estimation of unidentifiable links.

Fig. 6 (b)-(d) shows the path oscillations for Subito, UCB1, and BoundNT, respectively. Under stable traffic conditions, a routing algorithm should stabilize around a few paths to ensure a consistent and efficient utilization of network resources. Path oscillations can lead to unstable performance, poor resource utilization, thus affecting TCP flows' normal operation. However, Subito aims to prevent path switching and efficiently converge to the optimal path for any pair of monitors, minimizing disruptions and ensuring efficient data transmission. We measure path oscillations as the number of paths used by each approach, calculated over a moving window of 200 time slots.

Initially, Subito and UCB1 exhibit high oscillations due to their MAB nature, which prioritizes exploration versus exploitation at the beginning of the learning process. However, both approaches gradually converge to using less than five paths as they accumulate historical learned knowledge. Subito converges much faster than UCB1 and achieves a lower path oscillation, demonstrating its superior learning efficiency. Specifically, Subito finds the optimal path when the number of path oscillation converges to 1. As can be seen in Fig. 6 (b), this takes between 200 to 400 iterations, when the percentage of monitors is between 20% to 50%. In contrast, BoundNT maintains a steady path oscillation under all monitor settings. Furthermore, the path oscillations are higher when more monitors are available, mainly due to the lack of learning and the reliance on the latest delays. A path that appears to be the shortest at a certain iteration is likely not going to be at the next, leading to continuous switching and increasing path oscillations as more monitors, and thus paths, become available. These results demonstrate the strength of MAB learning in achieving efficient and stable routing compared to alternative approaches.

Experiment II: Scalability over network size in synthetic BA networks. In the second set of experiments, we scale the BA network size with 20, 40, 60, and 80 nodes and fix the percentage of monitors to 30%.



(a) (b)

Fig. 7. Experiment II: Average regret (a) and Monitoring overhead (b) increasing BA network size.

Fig. 7 (a) shows the average regret. For all approaches, learning becomes more difficult when the network is larger. Although the average regret increases for all approaches, Subito significantly outperforms UCB1 and BoundNT, resulting in a regret 60% less and closely comparable with Subito*. This achievement is attributed to the efficient learning and knowledge sharing among MAB agents, which are significantly beneficial capabilities when the network size grows. Unlike Subito, the agents of UCB1 learn at the path level and the knowledge is isolated from each other, which hinder learning efficiency and accuracy. Although BoundNT can benefit from its inference capability, it suffers again due to the lack of learning and only picks paths using the latest

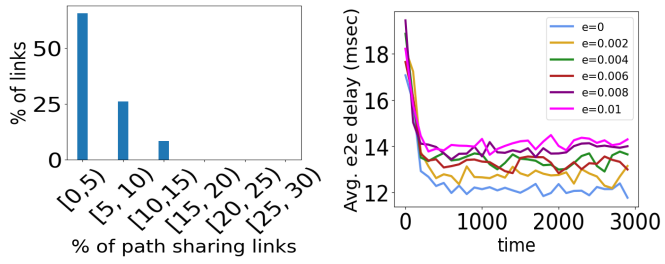
knowledge. This shortage deteriorates the performance and causes much higher additional delays with larger networks.

Fig. 7(b) shows the monitoring overhead. Subito and BoundNT provide similar performances and experience a slow increase as the network size grows. Conversely, UCB1 results in a surge of overhead from three times at network size 40 to almost six times at network size 80, compared to that of Subito* and BoundNT. These results validate the inference power of network tomography and its impressive capability of reducing the monitoring overhead.

Experiment III: Impact of shared links in synthetic BA network. In this set of experiments, we start evaluating the link distribution after Subito converges to the optimal paths. Fig. 8 (a) shows the percentage of links shared by multiple paths. It can be seen that links are well distributed. As a numerical example, no link is shared by more than 15% of the paths, and around 65% of the links are shared by less than 5% of the paths. These results support the evidence that Subito does not artificially create congestion by evenly distributing traffic in the network.

Subsequently, we study the impact of links shared by multiple paths in a BA network with 50 nodes, where 30% of nodes are selected as monitors. At each time slot t during the learning process, we increase the delay of a link depending on the number of paths going through that link. Specifically, given the expected delay d_k of each shared link l_k, assuming that n_k^t is the number of paths sharing link l_k at time t, the expected delay at time slot $t+1$ becomes $d_k + n_k^t \cdot e$, where e is a constant representing the contribution that each path adds to the delay.

Fig. 8 (b) shows the average end-to-end delay of the paths selected by Subito over time, under different settings of e, from 0 to 0.01 ms. In all considered scenarios, the delay quickly drops and stabilizes in a few iterations. This implies that, although the delays are affected by paths sharing the same links, Subito is able to converge towards a stable routing and learn the shortest path. It is worth noting that, the increase of average delays is due to higher values of e that affect the higher expected delays of links.

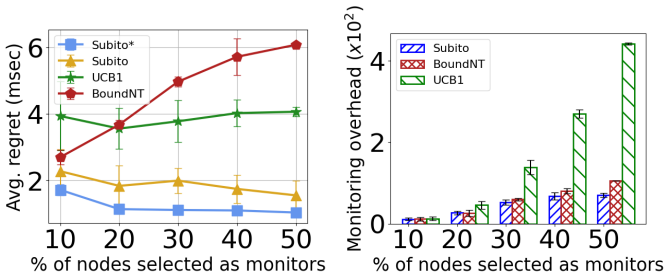


(a) (b)

Fig. 8. Experiment III: link utilization after subito converges to optimal paths (a) and End-to-end delay over time under different delay increments e (b).

D. Experiments Results in Real Networks

Experiment IV: Scalability over the percentage of monitors in BTN with synthetic dataset. Fig. 9 (a) compares the averaged regret among UCB1, BoundNT, Subito, and Subito*



(a) (b)

Fig. 9. Experiment IV: Average regret (a) and Monitoring overhead (b) increasing % of monitors in BTN.

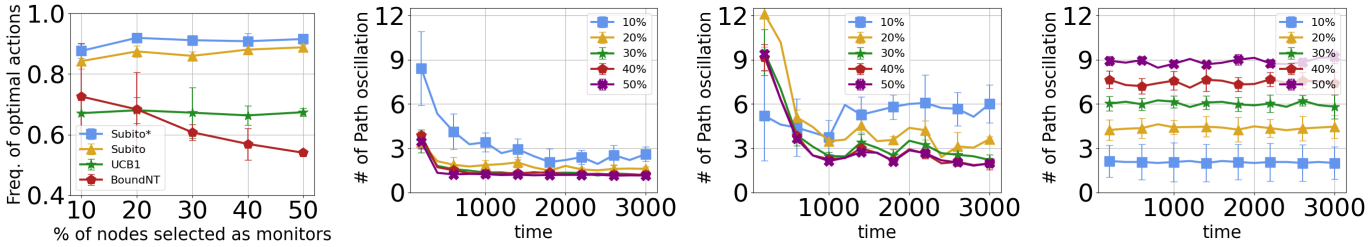
while increasing the number of monitors. Similar to the case of synthetic networks, BoundNT presents an increasing regret due to the lack of learning and higher number of available paths. UCB1 achieves smaller regret than BoundNT, but it is unable to fully exploit the benefit of extra monitors available. Conversely, Subito outperforms the comparison approaches and closely matches Subito* as the number of available monitors grows. By benefiting from the inference power of network tomography, and the shared knowledge among MAB agents, Subito achieves a regret that is two to three times lower than the comparison approaches.

Fig. 9 (b) shows the monitoring overhead as the monitor size increases. Subito outperforms BoundNT as the monitor size increases, being able to select a basis with smaller overhead. Conversely, UCB1 introduces monitoring overheads that are significantly larger (up to four times) than Subito and BoundNT. The result demonstrates again the benefit of combining the learning efficiency offered by MAB with the inference power provided *network tomography* in Subito.

Fig. 10 (a) shows the frequency of optimal actions while Fig. 10 (b)-(d) show the *path oscillation* of Subito, UCB1, and BoundNT, respectivelty. Once again, Subito significantly outperforms the comparison approaches. Subito shows an impressive ability to quickly learn the optimal paths as well as achieve very stable routing. Numerically, Subito has a frequency of optimal actions around 90% in all settings and uses less than 3 paths on average between monitors.

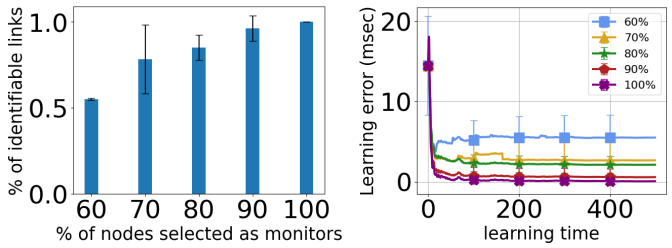
Experiment V: Scalability over the percentage of monitors in NSF with GENI testbed dataset. In this experiment, we investigate the impact of varying the number of monitors in a real NSF network using the dataset collected from the GENI testbed. we run the four approaches over 1500 time slots. Considering the relatively small size of the NSF network, the link identifiability is considerably low, with less than 5 nodes randomly selected as monitors. To ensure a fair and consistent comparison, we consider at least 60% of nodes serving as monitors.

Fig. 11 (a) illustrates the percentage of identifiable links as the number of monitors increases. With only 60% of nodes selected as monitors, the number of identifiable links is 54.5%, and it reaches 100% when monitors are deployed in each node. On the other hand, Fig. 11 (b) displays the corresponding learning error over the initial 600 time slots. As



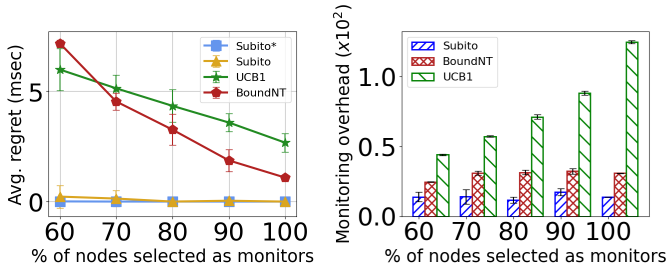
(a) (b) (c) (d)

Fig. 10. Experiment IV: Freq. of opt. actions (a) and path oscillations under Subito (b), UCB1 (c), and BoundNT (d) under different % of monitors in BTN.



(a) (b)

Fig. 11. Experiment V: Percentage of Identifiable Links (a) and Learning Error (b) varying the % of monitors using NSF network with real traffic data.



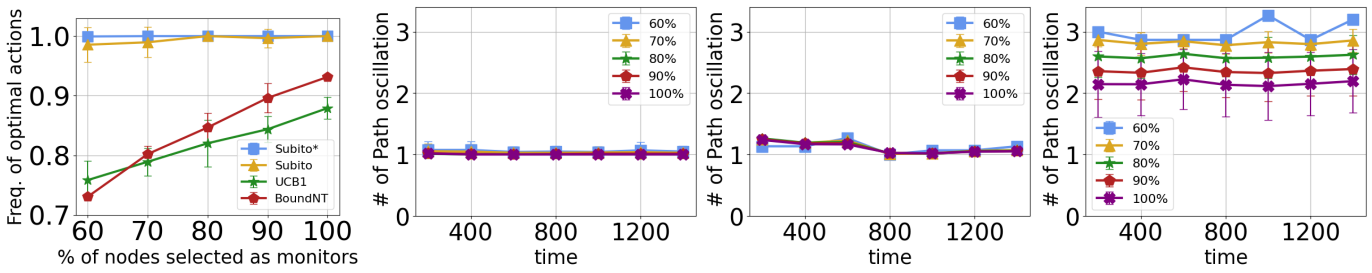
(a) (b)

Fig. 12. Experiment V: Average regret (a) and Monitoring overhead (b) increasing % of monitors in NSF network with real traffic data.

network tomography accurately infers link delay realizations for identifiable links, the error diminishes with the availability of more monitors and eventually converges to 0 when all links become identifiable with 100% monitors.

Fig. 12 (a) presents a comparison of the averaged regret among UCB1, BoundNT, Subito, and Subito* with an increasing number of monitors in the real network scenario. Unlike the cases with synthetic traffic data (Fig. 9(a)), BoundNT exhibits a decreasing regret in this scenario. In the synthetic dataset, the delay distribution X_k and the expected value d_k of each link l_k remains unchanged. However, at each time slot t, we randomly draw a realization of that distribution. Conversely, the flows in this NSF network are the actual traffic generated by the GENI testbed. This creates temporal dependencies, and smoother variability, between the delays over successive time slots. This characteristic aids BoundNT in achieving more accurate bound estimations for unidentifiable links, consequently exhibiting a decreasing regret owning to the smoother variability. Additionally, when the percentage of monitors exceeds 80%, more than 85% of links can be identified. BoundNT significantly benefits from the inference power of network tomography in these cases. UCB1 shows a significant regret decrease as the number of monitors increases. This is due to the relatively small size of this topology, which limits the path space to 2 paths per monitor pair. As the number of monitors increases, the shortest paths become shorter and more easily identifiable in such a small path space. Remarkably, Subito demonstrates exceptional performance in this experiment, achieving near-zero regrets even with only 60% monitors. This is due to the shared knowledge among agents, which enables in a smaller topology to learn the shortest paths in only few iterations. Numerically, Subito is able to converge to the optimal solutions only in around 15 to 20 iterations after the initialization phase. Fig. 12 (b) illustrates the monitoring overhead of the different approaches. Subito outperforms BoundNT as the number of monitors increases, enabling the selection of the optimal basis with the least overhead using Algorithm 2. It is worth noting that the overhead incurred by Subito reaches the number of links when the monitors reach 100%. This phenomenon is attributed to Subito's ability to reduce probing paths to a minimum basis, facilitated by Algo. 2, wherein each basis corresponds to the direct link connecting monitor pairs. Conversely, BoundNT blindly identifies a basis with no specific criteria, thus incurring a larger monitoring overhead. Conversely, UCB1 does not benefit from network topography and shared knowledge among agents. As a result, this approach introduces significantly larger monitoring overhead, up to four times that of BoundNT and eight times that of Subito.

We now assess the *quality* of routing under varying percentages of monitors. Fig. 13 (a) depicts the *frequency of optimal actions*, while (b)-(d) illustrate the *path oscillation* for Subito, UCB1, and BoundNT, respectively. Subito* achieves a near-optimal action rate of 1 with only 60% monitors, thanks to its perfect knowledge of link delays. Subito, despite having unknown delays, achieves similar performance as Subito*. This also results in a very low path oscillation. UCB1, with a limited path exploration space of 2, quickly converges to the optimal action, however at a slower rate than Subito. In contrast to synthetic traffic scenarios, BoundNT exhibits an improved rate of optimal actions and path oscillation this case. This is due to its reliance on network tomography and the similarity of link delays in consecutive time slots. However,



(a) (b) (c) (d)

Fig. 13. Experiment V: Freq. of opt. actions (a) and path oscillations under Subito (b), UCB1 (c), and BoundNT (d) under different % of monitors in NSF with real dataset.

path oscillation is still order of magnitudes higher than Subito and UCB1.

VI. Conclusion

In this study, we explore the application of network tomography for the purpose of a fundamental network management function, namely shortest-path routing, in SDN-based network architectures, which has not been investigated before. To achieve this, we introduce a novel approach named *Subito* that combines network tomography and reinforcement learning. Subito demonstrates superior performance in tackling the unique challenges associated with shortest-path routing in SDN environments and seamlessly integrates within an SDN controller. The proposed approach exhibits a bounded regret, minimizes monitoring overhead, and ensures stable routing. Through extensive experiments on both real and synthetic networks, we illustrate that Subito outperforms two state-of-the-art approaches across multiple performance metrics.

VII. Discussion and Future Work

In this paper we assumed that the southbound channels that allow communication between the SDN controller and monitors are dedicated links. If this assumption does not hold, and southbound channels are shared with the data links, Subito still presents several advantages discussed in the following. Subito is based on Network Tomography, which has been proposed with the exact purpose of reducing the monitoring overhead. Subito inherits such benefits. Specially, first, we do not need to monitor each link individually. We can rely on aggregated end-to-end measurements and exploit the power of network tomography to infer the delays of internal links. Furthermore, not all nodes need to act as monitors. Our experiments in section V show that even just 20% of the nodes acting as monitors can provide a very accurate view of the link delays. Additionally, we do not need to monitor all paths between monitor pairs, but only a subset of them that constitute a basis of the corresponding linear system. The clear advantage in terms of monitoring overhead are shown in Figures 7 (b) and 9 (b). In addition, Subito requires centralized communication between monitors and the SDN controller. This could be achieved by each monitor sending an individual packet to the SDN controller aggregating the information contained in the probing packets ending at that monitor along the probing path. As a result, the number of packets sent per round are *linear in the number of monitors*. It is important to highlight here that this is in contrast with standard Internet routing protocols, such as Link State Routing and OSPF. In the case of such protocols, each router sends a Link State packet to *all* other routers in the network. Without counting potential retransmissions of acknowledged flooding, the number of packets is *quadratic in the number of routers*. Noting that with Subito only 20% of nodes act as monitors, the advantages in terms of monitoring overhead are evident.

Furthermore, all routing algorithms need to provide reactivity to potential congestion or failures that may dynamically arise in the network. As a result, periodic monitoring is unavoidable to obtain fresh data on the status of network links, and any monitoring solution, e.g., via SNMP or ICMP messages, causes additional overhead to the network. Subito combines Network Tomography with reinforcement learning to reduce the monitoring overhead and quickly converge towards the shortest path routing. The frequency of end-to-end data collection and the corresponding communications with the SDN controller can be adjusted finding a tradeoff between reactivity and overhead. Ideally, such time could be even adjusted dynamically based on the characteristics of the traffic. We reserve the investigation on the setting of the monitoring period in our future work.

In our future work, we will also extend the proposed approach to deal with highly dynamic networks, where link delays learned by Subito may become outdated over time. This will require either a new learning process or an adaptive learning strategy. Furthermore, we will explore how the collection of other metrics, such as flow count and link bandwidth, can improve the learning and routing decisions. As an example, bandwidth tomography methodologies for reactive traffic engineering solutions can be further combined with multi-path techniques.

References

[1] G. Kakkavas, A. Stamou, V. Karyotis, and S. Papavassiliou, "Network Tomography for Efficient Monitoring in SDN-Enabled 5G Networks and Beyond: Challenges and Opportunities," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 70–76, 2021.

[2] C. Feng, L. Wang, K. Wu, and J. Wang, "Bound-based network tomography with additive metrics," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 316–324.

[3] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, "Inferring link metrics from end-to-end path measurements: Identifiability and monitor placement," *IEEE/ACM Transactions on Networking*, vol. 22, no. 4, pp. 1351–1368, 2014.

[4] Q. Zheng and G. Cao, "Minimizing probing cost and achieving identifiability in probe-based network link monitoring," *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 510–523, 2013.

[5] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An algebraic approach to practical and scalable overlay network monitoring," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 55–66, 2004.

[6] S. Tati, S. Silvestri, T. He, and T. La Porta, "Robust network tomography in the presence of failures," in *International Conference on Distributed Computing Systems*, 2014, pp. 481–492.

[7] R. Zhang, S. Newman, M. Ortolani, and S. Silvestri, "A network tomography approach for traffic monitoring in smart cities," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2268–2278, 2018.

[8] N. Bartolini, T. He, V. Arrigoni, A. Massini, F. Trombetti, and H. Khamfroush, "On fundamental bounds on failure identifiability by boolean network tomography," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 588–601, 2020.

[9] V. Arrigoni, N. Bartolini, A. Massini, and F. Trombetti, "Failure localization through progressive network tomography," in *IEEE Conference on Computer Communications (INFOCOM)*, 2021, pp. 1–10.

[10] V. Arrigoni and N. Bartolini, "Network identifiability: Advances in separating systems and networking applications," *IEEE Networking Letters*, vol. PP, pp. 1–1, 01 2022.

[11] T. He, L. Ma, A. Swami, and D. Towsley, *Network Tomography: Identifiability, Measurement Design, and Network State Inference*. Cambridge University Press, 2021.

[12] C.-C. Chiu and T. He, "Stealthy dgos attack against network tomography: The role of active measurements," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1745–1758, 2021.

[13] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data," in *ACM CoNEXT conference*, 2007, pp. 1–12.

[14] P. Zhang, Y. Zhao, Y. Wang, and Y. Jin, "SDN Enhanced Tomography for Performance Profiling in Cloud Network," *IET Communications*, vol. 11, no. 4, pp. 593–603, 2017.

[15] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 2211–2219.

[16] A. Sacco, M. Flocco, F. Esposito, and G. Marchetto, "Supporting Sustainable Virtual Network Mutations With Mystique," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2714–2727, 2021.

[17] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[18] D.-H. Le, H.-A. Tran, and S. Souihi, "A reinforcement learning-based solution for intra-domain egress selection," in *International Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1–6.

[19] J. Lee, *A first course in combinatorial optimization*. Cambridge University Press, 2004, vol. 36.

[20] Berman *et al.*, "GENI: A Federated Testbed for Innovative Network Experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

[21] C. Feng, L. Wang, K. Wu, and J. Wang, "Bound inference in network performance tomography with additive metrics," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1859–1871, 2020.

[22] B. Holbert, S. Tati, S. Silvestri, T. F. La Porta, and A. Swami, "Network topology inference with partial information," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 406–419, 2015.

[23] L. Xue, M. K. Marina, G. Li, and K. Zheng, "Paint: Path aware iterative network tomography for link metric inference," in *IEEE International Conference on Network Protocols(ICNP)*, 2022.

[24] Rkhami *et al.*, "On the Use of Machine Learning and Network Tomography for Network Slices Monitoring," in *IEEE 22nd International Conference on High Performance Switching and Routing (HPSR '21')*. IEEE, 2021, pp. 1–7.

[25] A. Sacco, F. Esposito, and G. Marchetto, "Completing and Predicting Internet Traffic Matrices Using Adversarial Autoencoders and Hidden Markov Models," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 2244–2258, 2023.

[26] A. Sacco, A. Angi, F. Esposito, and G. Marchetto, "HINT: Supporting Congestion Control Decisions with P4-driven In-Band Network Telemetry," in *IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2023, pp. 83–88.

[27] K. Liu and Q. Zhao, "Adaptive shortest-path routing under unknown and stochastically varying link states," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2012, pp. 232–237.

[28] A. Azzouni, R. Boutaba, and G. Pujolle, "Neuroute: Predictive dynamic routing for software-defined networks," in *International Conference on Network and Service Management (CNSM)*, 2017, pp. 1–6.

[29] A. Sacco, F. Esposito, and G. Marchetto, "RoPE: An Architecture for Adaptive Data-driven Routing Prediction at the Edge," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 986–999, 2020.

[30] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "Cfr-rl: Traffic engineering with reinforcement learning in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2249–2259, 2020.

[31] W. Xia, C. Di, H. Guo, and S. Li, "Reinforcement learning based stochastic shortest path finding in wireless sensor networks," *IEEE Access*, vol. 7, pp. 157 807–157 817, 2019.

[32] P. Zhou, J. Xu, W. Wang, Y. Hu, D. O. Wu, and S. Ji, "Toward optimal adaptive online shortest path routing with acceleration under jamming attack," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 1815–1829, 2019.

[33] J. Rischke, P. Sossalla, H. Salah, F. H. Fitzek, and M. Reisslein, "QR-SDN: Towards Reinforcement Learning States, Actions, and Rewards for Direct Flow Routing in Software-Defined Networks," *IEEE Access*, vol. 8, pp. 174 773–174 791, 2020.

[34] D. M. Casas-Velasco, O. M. C. Rendon, and N. L. da Fonseca, "Intelligent Routing Based on Reinforcement Learning for Software-Defined Networking," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 870–881, 2020.

[35] L. Zhao, J. Wang, J. Liu, and N. Kato, "Routing for Crowd Management in Smart Cities: A Deep Reinforcement Learning Perspective," *IEEE Communications Magazine*, vol. 57, no. 4, pp. 88–93, 2019.

[36] B. Dai, Y. Cao, Z. Wu, and Y. Xu, "IQoR-LSE: An Intelligent QoS On-Demand Routing Algorithm With Link State Estimation," *IEEE Systems Journal*, vol. 16, no. 4, pp. 5821–5830, 2022.

[37] X. Tao and S. Silvestri, "Network tomography and reinforcement learning for efficient routing," in *IEEE International Conference on Mobile Ad-Hoc and Smart Systems (MASS)*, 2023.

[38] Z. Hou, C. She, Y. Li, L. Zhuo, and B. Vucetic, "Prediction and communication co-design for ultra-reliable and low-latency communications," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 1196–1209, 2020.

[39] A. M. Sukhov and N. Y. Kuznetsova, "What type of distribution for packet delay in a global network should be used in the control theory?" *ArXiv*, vol. abs/0907.4468, 2009.

[40] H. Ortega-Arranz, A. Gonzalez-Escribano, and D. R. Llanos, *The shortest-path problem: Analysis and comparison of methods*. Springer Nature, 2022.

[41] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1466–1478, 2012.

[42] W. H. Greub, *Linear algebra*. Springer Science & Business Media, 2012, vol. 23.

[43] A.-L. Barabasi and R. Albert, "Albert, r.: Emergence of scaling in random networks. science 286, 509-512," *Science*, vol. 286, pp. 509–12, 11 1999.

[44] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan.

[45] D. L. Mills and H. Braun, "The nsfnet backbone network," in *Proceedings of the ACM Workshop on Frontiers in Computer Communications Technology*, ser. SIGCOMM '87. Association for Computing Machinery, 1987, p. 191–196.

[46] T. Dong, Q. Qi, J. Wang, A. X. Liu, H. Sun, Z. Zhuang, and J. Liao, "Generative Adversarial Network-based Transfer Reinforcement Learning for Routing With Prior Knowledge," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1673–1689, 2021.

Xu Tao received the M.Sc degree in computer engineering from Politecnico di Torino, Italy, in 2018. Then, she worked as a researcher in the Research Institute of LINKS Foundation, Italy, from 2018 to 2021. She is currently pursuing a Ph.D. degree in computer science at the University of Kentucky, USA. Her research lies in computer network monitoring and management; integration of IoT and cyber-physical systems in a variety of multidisciplinary areas, such as smart agriculture.

Doriana Monaco received the M.Sc degree in Computer Engineering from Politecnico di Torino, in 2022, where she is currently pursuing a Ph.D degree. Her research interests cover computer networks monitoring and management; distributed learning applications for Software-Defined Networks (SDN); cloud-computing solutions.

Alessio Sacco is an Assistant Professor at Politecnico di Torino. He received the M.Sc. degree (summa cum laude) and the Ph.D. degree (summa cum laude) in computer engineering from the Politecnico di Torino, Italy, in 2018 and 2022, respectively. His research interests include architecture and protocols for network management; implementation and design of cloud computing applications; algorithms and protocols for service-based architecture, such as Software Defined Networks (SDN), used in conjunction with Machine Learning algorithms.

Simone Silvestri (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer science from the Sapienza University of Rome, Italy, in 2006 and 2010, respectively. He is currently an Associate Professor with the Department of Computer Science, University of Kentucky. He has published more than 80 papers in international journals and conferences. His research interests include cyber–physical–human systems, the Internet of Things, energy management systems, terrestrial and aerial mobile networks, and network management. Dr. Silvestri's research is funded by several national and international agencies such as NIFA, NATO and the NSF, and he received the NSF CAREER award in 2020. He has served on the organizing committee of several international conferences, and the technical program committee of over 100 conferences.

Guido Marchetto received the Ph.D. degree in computer engineering from the Politecnico di Torino, in 2008, where he is currently an Associate Professor with the Department of Control and Computer Engineering. His research topics cover distributed systems and formal verification of systems and protocols. His interests also include network protocols and network architectures.