

# BRIGHT: A Bayesian Reinforcement Learning Framework for Intelligent Traffic Monitoring and Measurement

Hao Gong\* \*\*, Jun Chen\* \*\*, Xiaodong Zhu\* \*\*

\*School of Electronic, Electrical and Communication Engineering,  
University of Chinese Academy of Sciences, Beijing, China

\*\*National Network New Media Engineering Research Center, Institute of Acoustics,  
Chinese Academy of Sciences, Beijing, China

E-mail: gonghao23@mails.ucas.ac.cn; chenj@dsp.ac.cn; zhuxd@dsp.ac.cn

**Abstract**—Recent deployments of software-defined networking (SDN), especially OpenFlow, have enabled fine-grained traffic monitoring and traffic matrix (TM) estimation, yet existing methods still struggle under tight monitoring budgets in large networks due to three limitations: (1) lack of reasoning-driven adaptive monitoring plans that adjust intensity to traffic dynamics; (2) weak coupling between monitoring and inference, with little explicit per-flow uncertainty modeling for TM optimization; and (3) coarse spatio-temporal scheduling that ignores heterogeneous switch and link costs, causing excessive overhead and unbalanced loads. To address this, we propose BRIGHT, a Bayesian reinforcement-learning framework for intelligent traffic monitoring in OpenFlow networks. BRIGHT maintains lightweight Gaussian posteriors for per-flow rates and their uncertainty, uses stability and entropy indicators to categorize flows, and employs a linear contextual bandit with Bayesian Thompson sampling to adaptively select monitoring templates and per-slot measurement quotas. A spatio-temporal scheduler then maps each template to concrete probing intervals and low-overhead polling points along flow paths, achieving an online trade-off between monitoring cost and estimation accuracy. We implement BRIGHT as a Ryu controller application and evaluate it on a Mininet testbed with the GEANT backbone topology, comparing against LCM, AdaptMon, and OpenTM under stationary and bursty traffic. Experiments show that BRIGHT substantially reduces monitoring messages and control-plane load, improves effective link utilization, and still maintains or even improves TM estimation accuracy.

**Index Terms**—Software-Defined Networking (SDN), Traffic Monitoring, Bayesian Reinforcement Learning, Contextual Bandits.

## I. INTRODUCTION

Fine-grained traffic monitoring and measurement are fundamental to capacity planning, anomaly detection, quality-of-service (QoS) assurance, and traffic engineering in modern carrier and data-center networks [1], [2]. Traditional IP monitoring mechanisms such as NetFlow and sFlow [3], [4] let routers locally aggregate or sample packets and periodically export flow records, providing a global yet indirect view of network behaviour. However, these mechanisms are tightly

coupled to vendor-specific implementations and are not designed to support rapid, per-flow, network-wide adaptation under strict monitoring budgets [1].

Software-defined networking (SDN) decouples the control and data planes and exposes a logically centralised controller with a global view of the network [5]. In OpenFlow-based deployments, the controller can install flow rules and query per-flow counters on switches in a uniform way [6], enabling a shift from passive, device-centric statistics export to controller-driven traffic monitoring: the controller can decide which flows to poll, at which switches, and at what frequency. Yet naively polling all active flows at high frequency quickly becomes infeasible in large topologies due to the communication overhead on the control channel and the CPU load imposed on both controller and switches [7]. Obtaining an accurate view of network traffic under tight monitoring budgets therefore remains challenging.

A rich body of SDN monitoring frameworks addresses parts of this problem. One line of work, exemplified by OpenTM and subsequent flow-coverage schemes such as FlowCover and CeMon, exploits the global view of topology and active flows to select a subset of switches whose counters are sufficient to reconstruct the traffic matrix, often via combinatorial optimisation [8]–[11]. These approaches significantly reduce spatial monitoring cost but typically assume fixed or pre-configured polling periods and thus have limited ability to adapt to traffic dynamics. Another line of work, including PayLess and OpenNetMon, adapts polling frequency over time by adjusting measurement intervals according to application demands or observed variability, trading accuracy against overhead in the temporal domain [12]–[14]. Overall, most existing systems treat spatial and temporal decisions separately, rely on static heuristics or offline optimisation, and do not maintain an explicit probabilistic model of flow dynamics and estimation uncertainty. As a result, they can struggle to provide predictable accuracy under bursty traffic and tight monitoring budgets, and to automatically tune sampling intensity as network conditions evolve.

In parallel, reinforcement learning (RL) has been widely explored for SDN routing, resource allocation, and congestion

This work was supported by the Autonomous Deployment Project of the Institute of Acoustics, Chinese Academy of Sciences, “Research on Secure Transmission Technologies in Open Network Environments” (BSA202505).

control, where controllers learn to adjust paths or configuration parameters from performance feedback [15], [16]. However, these studies largely treat the measurement subsystem as a black box that provides sufficiently accurate state and focus on optimising control actions rather than monitoring itself. Actively deciding what to measure, when to measure, and where to measure under explicit accuracy–cost trade-offs therefore remains comparatively underexplored.

To address these limitations, this paper proposes *BRIGHT* (A Bayesian Reinforcement Learning Framework for Intelligent Traffic Monitoring and Measurement), which models SDN traffic monitoring as a sequential decision problem under explicit accuracy–cost constraints. At the data level, *BRIGHT* maintains a lightweight Gaussian posterior model for each flow and updates its belief whenever new counter samples are collected; posterior entropy and stability statistics derived from these models drive flow classification, prioritisation, and event detection. At the decision level, *BRIGHT* employs a linear contextual bandit with Bayesian Thompson sampling [17], [18] to select, at each decision epoch, from a small set of monitoring templates, each specifying a total monitoring budget and class-dependent base probing periods. A dedicated spatio-temporal scheduler refines these templates into per-flow probing intervals and chooses low-cost polling switches along each path, realising fine-grained allocation of limited monitoring resources.

Our main contributions are three-fold:

- **A unified Bayesian RL monitoring framework.** We propose *BRIGHT*, which combines per-flow Gaussian posterior modelling with a contextual bandit in a unified sequential decision framework that jointly orchestrates *what* to probe, *when* to probe, and *where* to probe in space and time.
- **An uncertainty-aware spatio-temporal scheduler.** We design a flow classification and prioritisation mechanism based on entropy and stability indicators, and use monitoring templates together with a spatio-temporal scheduler to map limited monitoring budgets to per-flow probing periods and low-cost polling points, enabling agile responses to bursty flows and strong compression for stable flows.
- **System implementation and evaluation on realistic topologies.** We implement *BRIGHT* as a northbound application on a Ryu controller and evaluate it on the GEANT backbone topology under both stationary and bursty traffic patterns, comparing against representative frameworks such as LCM, AdaptMon, and OpenTM. Our experiments demonstrate that *BRIGHT* reduces monitoring overhead, improves effective link utilisation, and lowers control-plane latency while maintaining or improving traffic-matrix estimation accuracy.

## II. RELATED WORK

In OpenFlow-based SDNs, many systems have been proposed to estimate traffic matrices and monitor flows with

reduced overhead. OpenTM is an early traffic-matrix estimator that exploits per-flow counters in OpenFlow switches and routing information in the controller to directly measure origin–destination traffic, selecting a subset of switches along each path to query so as to reduce statistics messages and switch load [8]. Subsequent flow-coverage schemes such as FlowCover, CeMon, and related SDN-based TM estimation methods generalise this idea by formulating switch selection as a coverage or optimisation problem using the controller’s global view, thereby further lowering spatial monitoring cost [9]–[11]. However, these approaches largely target spatial coverage under relatively stable sets of flows and paths and do not explicitly track monitoring delay or the evolution of flow-level estimation error over time.

A complementary line of work emphasizes temporal adaptation. PayLess introduces a low-cost monitoring framework that supports multiple aggregation levels and refresh periods, and adjusts polling intervals according to inferred traffic dynamics in order to improve the timeliness and accuracy of global metrics under limited control-plane bandwidth [12]. OpenNetMon focuses on QoS monitoring and provides per-flow measurements of throughput, delay, and packet loss for flows traversing edge switches, together with an open-source implementation for OpenFlow controllers [13]. More recent schemes based on recursive learning further tune sampling intervals according to observed traffic patterns [14]. These methods, however, rely primarily on rule- or threshold-based logic and do not jointly optimise where and how often to poll within a unified cost or error model.

To explicitly incorporate reporting latency and capacity constraints, several optimisation-based approaches treat flow–switch assignment and polling as constrained optimisation problems. A representative study formulates the problem of minimising flow monitoring cost under per-flow delay and capacity constraints as an integer linear program and proposes heuristic algorithms such as Low-Cost Monitoring (LCM) [19]. Such methods are well suited to longer time-scale resource planning but assume fixed delay requirements and flow sets, and they do not maintain per-flow posterior models or update monitoring decisions at fine time scales as estimation quality changes. Transition-enabled monitoring services such as AdaptMon instead orchestrate controlled switches between multiple monitoring mechanisms to better match changing network conditions and application requirements [20]. While this improves robustness at a macroscopic level, the internal polling logic of each mechanism is typically predefined, and there is no unified flow-level scheduler that jointly considers temporal sampling, spatial placement, and resource constraints driven by measurement uncertainty. In contrast, the *BRIGHT* framework proposed in this work explicitly maintains Gaussian posterior state at the flow level and uses entropy and stability as first-class signals in a unified decision process that coordinates sampling templates and spatio-temporal allocation of limited monitoring resources.

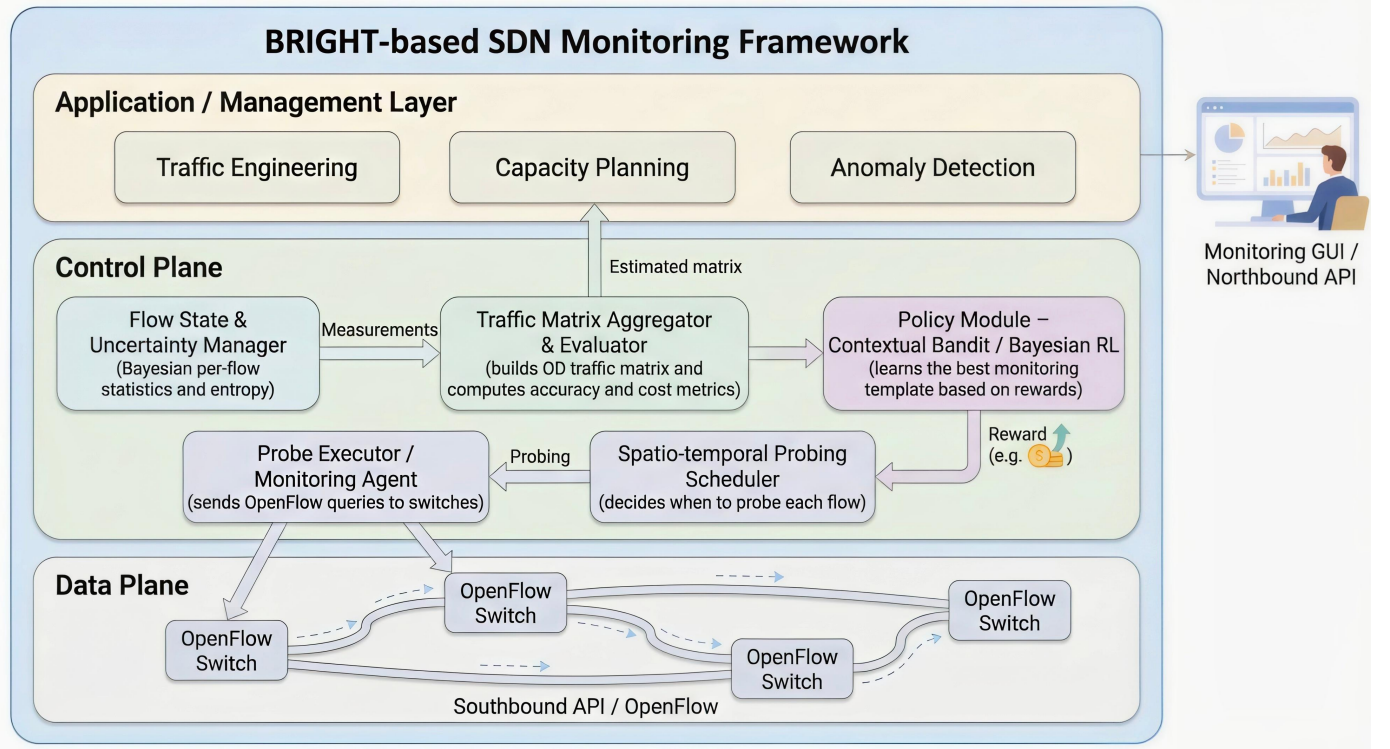


Fig. 1. Overall BRIGHT-based SDN monitoring framework. The control plane hosts the Flow State & Uncertainty Manager, Traffic Matrix Aggregator & Evaluator, Policy Module (contextual bandit / Bayesian RL), Spatio-temporal Probing Scheduler, and Probe Executor / Monitoring Agent, while applications in the management layer consume the estimated OD traffic matrix and OpenFlow switches in the data plane provide per-flow measurements.

### III. METHOD

In this section, we present BRIGHT, a Bayesian reinforcement learning framework for intelligent traffic monitoring and measurement in OpenFlow-based SDNs.

#### A. Problem Formulation

We consider a single-controller OpenFlow network where the controller maintains the global topology and routing information, and thus knows the path of each active flow. Time is divided into equal-length slots of duration  $\Delta t$ , indexed by  $t = 1, 2, \dots$

A unidirectional flow  $f$  is uniquely identified by the 5-tuple (srcIP, dstIP, proto, srcPort, dstPort). Let  $r_f(t)$  denote the true average rate of flow  $f$  in time slot  $t$ . For any origin-destination (OD) pair  $(i, j)$ , the corresponding traffic-matrix element is

$$R_{ij}(t) = \sum_{f \in \mathcal{F}_{i \rightarrow j}} r_f(t), \quad (1)$$

where  $\mathcal{F}_{i \rightarrow j}$  is the set of flows from  $i$  to  $j$ . Due to limited control-plane bandwidth and switch/controller CPU capacity, the controller cannot measure all flows in every slot, but must select a subset of flow-switch pairs to poll.

If the controller queries a counter for flow  $f$  at some switch along its path in slot  $t$ , it obtains the total bytes  $B_f(t)$  accumulated during that slot and forms the observation

$$y_f(t) = \frac{B_f(t)}{\Delta t}, \quad (2)$$

which is treated as a noisy estimate of the true rate  $r_f(t)$ .

To quantify monitoring overhead, we assign each switch  $sw$  a query cost  $\text{cost}(sw)$ , capturing, e.g., the number of statistics messages or monitoring bandwidth consumed when polling that switch. Let  $Q_t$  be the set of switches polled in slot  $t$ ; the total monitoring cost is then

$$\text{Cost}_t = \sum_{sw \in Q_t} \text{cost}(sw). \quad (3)$$

The objective of BRIGHT is to obtain accurate estimates of the OD traffic matrix  $\{R_{ij}(t)\}$  while keeping the long-term average monitoring cost small, and to adapt gracefully when the traffic pattern transitions between stationary and bursty regimes.

#### B. Overview and Architecture

BRIGHT is deployed on top of the SDN controller and adopts a three-layer design, illustrated in Fig. 1.

**Application / Management Layer:** This layer includes traffic engineering, capacity planning, anomaly detection, and other management applications. They consume the OD traffic matrix estimates and monitoring statistics produced by BRIGHT via a northbound API or monitoring GUI to drive network-level decisions.

**Control Plane:** The control plane hosts the core logic of BRIGHT and consists of five modules:

- **Flow State & Uncertainty Manager:** maintains per-flow Bayesian state and uncertainty metrics;

**Algorithm 1** BRIGHT Online Monitoring and Learning

**Require:** Network topology  $\mathcal{G}$ , routing, slot length  $\Delta t$ , initial priors, budget parameters

**Ensure:** Traffic-matrix estimates  $\{\hat{R}_{ij,t}\}$  and learned policy parameters

- 1: Initialize per-flow priors and stability indicators; initialize bandit statistics  $A_0, b_0$
- 2: **for**  $t = 1, 2, \dots$  **do**
- 3:   **Context construction:** derive global statistics from current per-flow posteriors  $(\mu_f(t), \sigma_f^2(t))$  and stability  $S_f(t)$ , and form context  $x_t$
- 4:   **Policy selection:** run BLTS with context  $x_t$  to sample arm  $a_t$ ; obtain class-wise base periods  $T_{\text{base}}[c, a_t]$  and budget  $K(a_t)$
- 5:   **Scheduling:** compute per-flow probing periods  $T_{\text{probe},f}$  from  $T_{\text{base}}[c(f), a_t]$  and  $S_f(t)$ ; identify due flows and select probing set  $\mathcal{F}_t$  under  $K(a_t)$  and cost constraints
- 6:   **Probe execution:** for each  $f \in \mathcal{F}_t$ , select a low-cost polling switch on path  $f$  subject to capacity; query counters, obtain  $B_f(t)$  and observation  $y_f(t) = B_f(t)/\Delta t$
- 7:   **Flow state update:** update per-flow posteriors  $(\mu_f(t), \sigma_f^2(t))$  and stability  $S_f(t)$  for probed flows using  $\{y_f(t)\}$
- 8:   **Evaluation:** aggregate  $\mu_f(t)$  to OD-level TM  $\hat{R}_{ij,t}$ ; compute NRMSE<sub>t</sub>, monitoring cost Cost<sub>t</sub> and load variance Var(load<sub>t</sub>); form reward  $r_t = -(\text{NRMSE}_t + \lambda_1 \text{Cost}_t + \lambda_2 \text{Var}(\text{load}_t))$
- 9:   **Bandit update:** update  $A_{t+1}, b_{t+1}$  using  $(x_t, a_t, r_t)$
- 10: **end for**

- **Traffic Matrix Aggregator & Evaluator:** constructs OD traffic matrices from per-flow estimates and computes accuracy, cost, and load-balancing metrics;
- **Policy Module (Contextual Bandit / Bayesian RL):** learns and selects monitoring templates based on contextual information and evaluation feedback;
- **Spatio-temporal Probing Scheduler:** allocates concrete probing periods and priorities to individual flows under the chosen template;
- **Probe Executor / Monitoring Agent:** sends OpenFlow statistics requests to switches and collects counter samples.

*Data Plane:* The data plane consists of OpenFlow switches that receive flow rules and statistics queries from the controller via the southbound API, maintain per-flow counters locally, and return measurement results.

In each time slot, BRIGHT executes a closed-loop procedure:

- 1) *Probe execution:* the Probe Executor issues OpenFlow statistics requests to selected switches according to the plan from the Spatio-temporal Probing Scheduler and the available budget, and gathers counter samples.
- 2) *State update:* the Flow State & Uncertainty Manager updates the Gaussian posterior for each probed flow

using the new observations and refreshes uncertainty and flow-category statistics.

- 3) *Aggregation and evaluation:* the Traffic Matrix Aggregator & Evaluator aggregates per-flow estimates into an OD traffic matrix, computes error, monitoring cost, and load-balancing metrics, and exposes these to the application layer while also forming a reward signal.
- 4) *Policy update:* the Policy Module updates its contextual bandit model with the current context and reward, and selects a monitoring template for the next slot.
- 5) *Spatio-temporal scheduling:* the Spatio-temporal Probing Scheduler refines the selected template into per-flow probing periods and polling locations, preparing the probing plan for the next slot.

This loop couples per-flow Bayesian estimation with policy learning and spatio-temporal scheduling, enabling BRIGHT to adaptively balance monitoring overhead and estimation accuracy. The overall online monitoring and learning loop of BRIGHT is summarised in Algorithm 1.

### C. Flow State and Uncertainty Management

The Flow State & Uncertainty Manager maintains, for each active flow  $f$ , a one-dimensional Gaussian posterior

$$r_f \sim \mathcal{N}(\mu_f, \sigma_f^2), \quad (4)$$

together with a stability indicator  $S_f$  and an event flag. Whenever flow  $f$  is probed in slot  $t$  and an observation  $y_f(t)$  is obtained, we assume additive Gaussian noise and update  $\mu_f$  and  $\sigma_f^2$  using the standard Bayesian update equations. The resulting posterior  $(\mu_f, \sigma_f^2)$  is used both to construct OD traffic-matrix estimates and to provide uncertainty information to downstream modules.

Uncertainty is quantified via the differential entropy of the posterior:

$$H_f = \frac{1}{2} \ln(2\pi e \sigma_f^2). \quad (5)$$

In addition, BRIGHT computes a stability score  $S_f \in [0, 1]$  by normalising the sample variance of recent observations within a sliding window; higher  $S_f$  indicates more volatile short-term behaviour. Combining  $\mu_f$ ,  $H_f$ , and the deviation between current observations and the historical mean, flows are categorised into four classes: stable elephant flows, stable mouse flows, unstable flows, and event flows. Each category is assigned different default probing periods and priorities in later stages.

At the end of each slot, the manager aggregates global statistics such as average entropy and the fraction of unstable/event flows, and combines them with the previous slot's traffic-matrix error and monitoring cost to form a context vector  $x_t$  for the Policy Module.

### D. Policy Module: Linear Contextual Bandit

The Policy Module controls the global sampling intensity via a finite set of *monitoring templates* (arms). Let  $x_t$  denote the context vector in slot  $t$ , including, e.g., the previous slot's traffic-matrix error, average uncertainty, the proportion

of unstable/event flows, and the ratio between actual cost and budget. Each arm  $a \in \mathcal{A}$  specifies class-dependent base probing periods  $T_{\text{base}}[c, a]$  and a maximum number of flows  $K(a)$  that may be probed in a slot.

For any context-arm pair  $(x_t, a)$ , we construct a feature vector  $\phi(x_t, a)$  and model the expected reward as

$$r_t = \theta^\top \phi(x_t, a_t) + \varepsilon_t, \quad (6)$$

where  $\theta$  is an unknown parameter vector and  $\varepsilon_t$  is zero-mean noise. BRIGHT employs linear Thompson sampling with a Gaussian prior over  $\theta$ , maintaining sufficient statistics  $(A_t, b_t)$  for the posterior.

At the beginning of slot  $t$ , the Policy Module samples a parameter vector  $\theta_t$  from the current posterior, computes a stochastic estimate  $\tilde{r}_{t,a} = \theta_t^\top \phi(x_t, a)$  for each arm, and selects

$$a_t = \arg \max_{a \in \mathcal{A}} \tilde{r}_{t,a}. \quad (7)$$

The chosen arm determines the total monitoring budget and class-level base probing periods for slot  $t$ . After the slot finishes and the reward  $r_t$  is computed by the evaluator, the module updates  $(A_{t+1}, b_{t+1})$ . This procedure enjoys the sublinear regret guarantees of linear contextual bandits while gradually converging to high-quality monitoring template choices.

#### E. Spatio-temporal Scheduling and Probe Execution

Given the selected arm  $a_t$ , the Spatio-temporal Probing Scheduler assigns concrete probing periods to individual flows and selects which flows to probe in the current slot under the available budget.

For a flow  $f$  with category  $c(f)$  and base period  $T_{\text{base}}[c(f), a_t]$ , BRIGHT adjusts the period using a proportional controller based on its stability score:

$$T_{\text{probe},f} = \text{clip}\left(\frac{T_{\text{base}}[c(f), a_t]}{1 + K_p(1 - S_f/S_{\text{target}})}, T_{\min}, T_{\max}\right), \quad (8)$$

where  $K_p$  is a gain parameter,  $S_{\text{target}}$  is a target stability level, and  $\text{clip}(\cdot)$  truncates the period to the range  $[T_{\min}, T_{\max}]$ . Flows with poor stability ( $S_f$  large) receive shorter probing periods, whereas highly stable flows are probed more sparsely.

If  $t - T_{\text{update},f} \geq T_{\text{probe},f}$ , flow  $f$  is considered *due* in slot  $t$ . For all due flows, the scheduler computes a composite priority score combining normalised entropy  $\tilde{H}_f$ , temporal urgency, and the event flag. For example,

$$\text{Urgency}_f = \frac{t - T_{\text{update},f}}{T_{\text{probe},f}}, \quad (9)$$

$$\text{Score}_f = w_1 \tilde{H}_f + w_2 \text{Urgency}_f + w_3 \cdot \text{event}_f, \quad (10)$$

where  $w_1, w_2, w_3$  are tunable weights and  $\text{event}_f \in \{0, 1\}$  indicates detected anomalies. Under the slot-level budget  $K(a_t)$  and optional cost constraints, the scheduler sorts due flows by  $\text{Score}_f$  in descending order and selects the top ones to form the probing set  $\mathcal{F}_t$ .

For each flow  $f \in \mathcal{F}_t$ , the Probe Executor / Monitoring Agent examines its path  $\text{path}_f = [\text{sw}_1, \dots, \text{sw}_L]$  and chooses

a polling switch with remaining capacity and minimum query cost:

$$\text{sw}^* = \arg \min_{\text{sw} \in \text{Cand}_f} \text{cost}(\text{sw}), \quad (11)$$

where  $\text{Cand}_f$  is the set of feasible switches for flow  $f$ . The executor then sends an OpenFlow statistics request to  $\text{sw}^*$ , obtains the counter increment  $B_f(t)$ , and forms the observation  $y_f(t)$ . It also updates the monitoring load on each switch and the total cost  $\text{Cost}_t$ , which will be used in the evaluation step.

#### F. Traffic Matrix Aggregation, Evaluation, and Closed-Loop Learning

At the end of slot  $t$ , the Traffic Matrix Aggregator & Evaluator uses the posterior means  $\mu_f(t)$  to obtain an OD traffic matrix estimate:

$$\hat{R}_{ij,t} = \sum_{f \in \mathcal{F}_{ij}} \mu_f(t), \quad (12)$$

where  $\mathcal{F}_{ij}$  is the set of flows from  $i$  to  $j$ . In simulation or trace-driven experiments, the corresponding ground truth  $R_{ij,t}$  is available, enabling computation of a normalised root-mean-square error (NRMSE)  $\text{NRMSE}_t$  as an accuracy metric. The evaluator also aggregates the total monitoring cost  $\text{Cost}_t$  and the variance of monitoring load across switches,  $\text{Var}(\text{load}_t)$ . The estimated traffic matrix and these metrics are exported to the application layer via the northbound interface.

To drive reinforcement learning, BRIGHT combines accuracy, cost, and load balancing into a scalar reward:

$$r_t = -(\text{NRMSE}_t + \lambda_1 \text{Cost}_t + \lambda_2 \text{Var}(\text{load}_t)), \quad (13)$$

where  $\lambda_1$  and  $\lambda_2$  control the relative importance of accuracy, monitoring overhead, and load balancing. Finally, the triplet  $(x_t, a_t, r_t)$  is passed back to the Policy Module to update the posterior of the linear contextual bandit. By repeating this process, BRIGHT continually adapts monitoring templates and sampling strategies to evolving network conditions, achieving an adaptive balance between monitoring cost and traffic-matrix estimation accuracy.

### IV. EXPERIMENTAL

#### A. Network Topology and Traffic

We emulate a wide-area backbone based on the GÉANT topology from the SNDlib library, which provides realistic carrier-grade network instances for research on survivable fixed telecommunication networks [21]. The GÉANT backbone is modeled as a directed graph with 23 nodes and 74 directed links (37 bidirectional links) [22]. The resulting topology is illustrated in Fig. 2.

#### B. Traffic Models

We fix 100 active OD flows on the GEANT topology and model their rates on a slotted time axis. Each flow  $f$  is marked as an elephant with probability  $p_{\text{ele}} = 0.2$  and as a mouse otherwise. Its baseline rate  $\mu_f$  is drawn from

$$\mu_f \sim \text{Uniform}[50, 100] \text{ Mbps for elephants,}$$

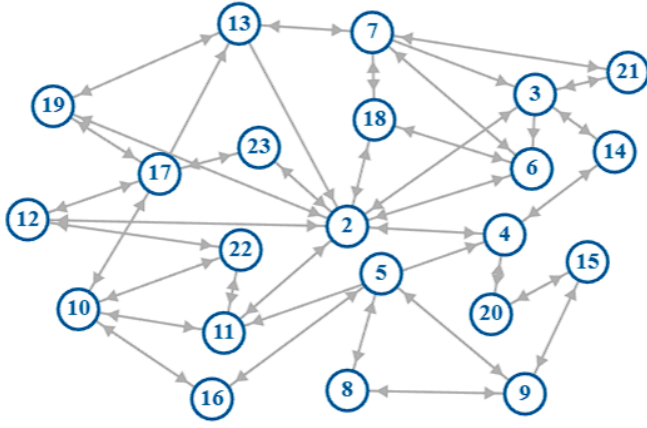


Fig. 2. GEANT backbone network topology (23 nodes and 74 directed links), constructed from the GEANT network instance provided by SNDlib.

and

$$\mu_f \sim \text{Uniform}[5, 20] \text{ Mbps for mice.}$$

1) *Stationary pattern*: Each flow fluctuates mildly around its baseline,

$$r_f(t) = \mu_f(1 + \xi_f(t)), \quad (14)$$

where  $\xi_f(t)$  is a zero-mean noise term drawn from a truncated Gaussian  $\mathcal{N}(0, \sigma_{\text{stat}}^2)$ , clipped to  $[-0.3, 0.3]$  (typically  $\sigma_{\text{stat}} = 0.05$ ), optionally generated by a first-order AR process. This pattern contains no explicit bursts and is used to assess long-term overhead and accuracy under statistically stationary load.

2) *Bursty pattern*: For the bursty scenario, a fraction  $p_{\text{burst}} \approx 0.3$  of flows are randomly chosen as burst candidates. Each candidate has a discrete state  $s_f(t) \in \{N, H, L\}$  (normal, high, low) and uses the same noise process  $\xi_f(t)$  as in (14), with

$$r_f(t) = \begin{cases} \mu_f(1 + \xi_f(t)), & s_f(t) = N, \\ \alpha_H \mu_f(1 + \xi_f(t)), & s_f(t) = H, \\ \alpha_L \mu_f(1 + \xi_f(t)), & s_f(t) = L, \end{cases} \quad (15)$$

where  $\alpha_H = 3$  and  $\alpha_L = 0.3$ . The state  $s_f(t)$  follows a three-state Markov chain: from  $N$  the flow enters  $H$  or  $L$  with probabilities  $p_{\text{up}} = p_{\text{down}} = 0.01$ , and from  $H$  or  $L$  it returns to  $N$  with probabilities  $q_H = q_L = 0.3$ .

### C. Implementation and Baselines

All experiments are conducted on a commodity Linux server running the Mininet network emulator [23]. We use the open-source Ryu SDN framework as the controller platform [24], [25], and implement BRIGHT and all baseline algorithms as separate Ryu applications. Unless otherwise stated, the controller maintains shortest-path routes on the GEANT topology, and time is discretized into slots of length  $\Delta t$  (e.g.,  $\Delta t = 5$  s); each method operates in the same slot-based loop, deciding which flows to probe and issuing OpenFlow FLOW\_STATS requests.

We compare BRIGHT against three representative monitoring schemes introduced in Section II, namely *LCM*, *AdaptMon*,

and *OpenTM*. All methods run on the same controller and topology under identical traffic traces and random seeds to ensure a fair comparison.

### D. Evaluation Metrics

To compare BRIGHT with the baselines, we adopt four metrics covering control-plane signaling, data-plane resource usage, and controller latency. For each metric, we introduce a unified notation.

1) *Monitoring Message Count*: Let  $M_t^{\text{mon}}$  denote the number of monitoring-related messages in slot  $t$ , including FLOW\_STATS\_REQUEST and FLOW\_STATS\_REPLY generated by the monitoring application. For each method, we consider only the total number of monitoring messages over the whole experiment:

$$M_{\text{tot}}^{\text{mon}} = \sum_{t=1}^T M_t^{\text{mon}}. \quad (16)$$

This metric reflects the overall scale of statistics operations triggered by the monitoring strategy.

2) *Monitoring-Aware Network Utilization*: To distinguish between user traffic and monitoring traffic, we let  $L_\ell^{\text{data}}(t)$  and  $L_\ell^{\text{mon}}(t)$  denote, respectively, the data and monitoring loads on link  $\ell$  in slot  $t$ , and let  $C_\ell$  denote its capacity. We define the data and monitoring utilizations as

$$U_\ell^{\text{data}}(t) = \frac{L_\ell^{\text{data}}(t)}{C_\ell}, \quad U_\ell^{\text{mon}}(t) = \frac{L_\ell^{\text{mon}}(t)}{C_\ell}. \quad (17)$$

We are interested in two aggregate quantities. First, the per-slot peak monitoring utilization on the most heavily loaded link:

$$U_{\text{max}}^{\text{mon}}(t) = \max_{\ell} U_\ell^{\text{mon}}(t), \quad (18)$$

which characterizes how severely monitoring traffic occupies potential bottleneck links. Second, the overall *effective utilization* of link capacity by user traffic:

$$\text{EffUtil} = \frac{\sum_{\ell, t} L_\ell^{\text{data}}(t)}{\sum_{\ell, t} (L_\ell^{\text{data}}(t) + L_\ell^{\text{mon}}(t))}, \quad (19)$$

which measures what fraction of total link resources over time is used for real data rather than monitoring overhead.

3) *Control-Plane Latency*: We use a single control-plane round-trip time metric  $T_{\text{cp}}$  to quantify the impact of monitoring on controller responsiveness. While a given monitoring method is running, we periodically inject probe control events (e.g., Packet-In / FlowMod exchanges at a fixed frequency) and measure the elapsed time from sending a probe event to receiving the corresponding controller response. Let  $T_{\text{cp}}^{(k)}$  denote the  $k$ -th observed round-trip time, for  $k = 1, \dots, K$ . For each method, we compute the average

$$\bar{T}_{\text{cp}} = \frac{1}{K} \sum_{k=1}^K T_{\text{cp}}^{(k)} \quad (20)$$

to characterize tail latency. Heavier monitoring leads to more congestion in the control plane and thus larger  $T_{\text{cp}}^{(k)}$ .

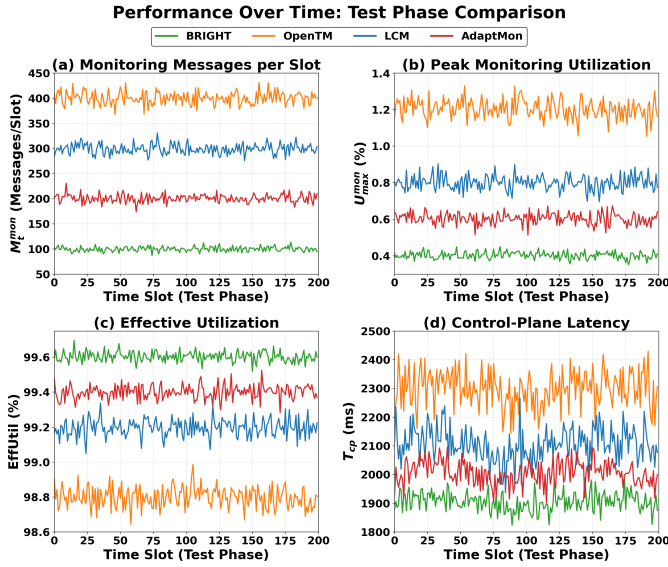


Fig. 3. Performance of different algorithms under the Stationary Pattern scenario; subfigures (a)–(d) correspond to the experimental metrics used in our evaluation.

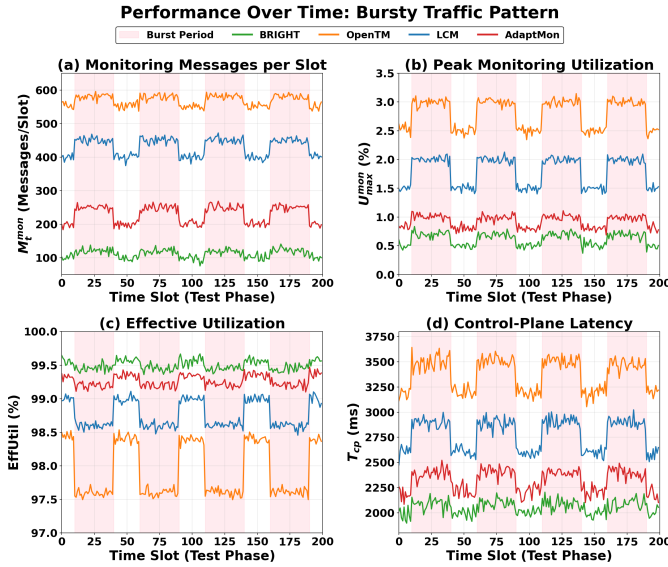


Fig. 4. Performance of different algorithms under the Bursty Pattern scenario; subfigures (a)–(d) correspond to the experimental metrics used in our evaluation.

## E. Main Result

1) *Experimental Results and Discussion:* Fig. 4 and Fig. 3 report the evolution of the four performance metrics over time under, respectively, the bursty traffic pattern and the stationary test phase. Across both settings, BRIGHT consistently outperforms OpenTM, LCM, and AdaptMon in terms of monitoring overhead, link utilisation, and control-plane latency.

2) *Monitoring overhead and monitoring utilisation:* In both traffic patterns, BRIGHT achieves the lowest number of monitoring messages per slot  $M_t^{\text{mon}}$ . During the stationary test phase, BRIGHT stays around one hundred messages per

slot, whereas LCM and AdaptMon remain in the few-hundred range and OpenTM approaches or exceeds 400 messages/slot. Under bursty traffic (Fig. 4), BRIGHT moderately increases its probing rate during the shaded burst intervals, but its overhead remains clearly below all baselines; by contrast, OpenTM exhibits almost no adaptation to bursts, and AdaptMon maintains a persistently high probing rate. This behaviour is mirrored in the peak monitoring utilisation  $U_{\text{mon}}^{\text{max}}$ , where BRIGHT achieves the lowest values over time, indicating that it preserves more bandwidth on bottleneck links for data traffic.

3) *Effective utilisation and control-plane latency:* Because monitoring traffic is more economical, BRIGHT attains the highest effective utilisation (EffUtil) in both scenarios, i.e., the largest fraction of link capacity is used by data traffic rather than monitoring packets. OpenTM, which generates the most monitoring traffic, yields the lowest EffUtil. At the same time, BRIGHT delivers the smallest and most stable control-plane latency  $T_{\text{cp}}$ . In the bursty setting, its latency only increases slightly during burst intervals, whereas LCM and OpenTM exhibit pronounced latency spikes, and AdaptMon also remains above BRIGHT overall. This shows that BRIGHT can raise monitoring intensity when needed without overloading the controller or switches.

4) *Sources of BRIGHT's advantages:* These gains stem from three key design choices in BRIGHT: (i) *Bayesian per-flow modelling and uncertainty awareness:* the Flow State & Uncertainty Manager maintains posterior means and variances for each flow and derives entropy and stability indicators, allowing the scheduler to focus probes on uncertain, abnormal, and bursty flows while aggressively compressing stable flows, thereby eliminating redundant polling; (ii) *Contextual bandit-based strategy learning:* the Policy Module selects monitoring templates based on traffic-matrix error, global uncertainty, and cost-to-budget ratio, automatically switching between aggressive and conservative monitoring regimes so that probing intensity is increased during bursts and throttled during steady periods; and (iii) *Spatio-temporal joint scheduling with cost-aware probing:* the Spatio-temporal Probing Scheduler assigns flow-specific probing periods according to flow class and stability, while the Probe Executor chooses low-cost, non-saturated switches along each path, concentrating the available budget on the most informative locations and avoiding monitoring hot-spots.

Overall, these results demonstrate that BRIGHT's closed-loop design—combining Bayesian state estimation, reinforcement-learning-based template selection, and spatio-temporal scheduling—can significantly reduce monitoring overhead, improve effective link utilisation, and lower control-plane latency, while preserving accurate traffic-matrix estimates.

## V. CONCLUSION

This paper presented BRIGHT, a Bayesian reinforcement learning framework for intelligent traffic monitoring and measurement in OpenFlow-based SDNs. Unlike existing solutions that separately optimise spatial coverage or temporal sampling

with fixed heuristics, BRIGHT treats monitoring as a sequential decision problem under explicit accuracy–cost constraints. At the data level, it maintains Bayesian per-flow models and derives entropy- and stability-based uncertainty indicators; at the decision level, it combines a contextual bandit policy module with a spatio-temporal probing scheduler and a cost-aware probe executor to jointly decide what to probe, when to probe, and where to probe.

We implemented BRIGHT as a northbound application on a Ryu controller and evaluated it on the GEANT backbone topology under both stationary and bursty traffic patterns against OpenTM, LCM, and AdaptMon. The results show that BRIGHT substantially reduces monitoring messages and peak monitoring utilisation, improves effective link utilisation, and lowers control-plane latency, while maintaining or improving traffic-matrix estimation accuracy. These gains highlight the advantage of integrating Bayesian inference with reinforcement-learning-based strategy selection for SDN monitoring systems.

Future work includes extending BRIGHT to multi-controller deployments, exploring alternative reward designs and richer uncertainty metrics, and validating the framework in larger-scale testbeds and production environments, including specific experiments on both ONOS and OpenDaylight (ODL) controllers to evaluate cross-platform compatibility and performance characteristics.

#### ACKNOWLEDGMENT

This work was supported by the Autonomous Deployment Project of the Institute of Acoustics, Chinese Academy of Sciences, “Research on Secure Transmission Technologies in Open Network Environments” (BSA202505). Corresponding author: Xiaodong Zhu.

#### REFERENCES

- [1] M. S. Islam, M. Al-Mukhtar, M. R. K. Khan, and M. Hossain, “A Survey on SDN and SDCN Traffic Measurement: Existing Approaches and Research Challenges,” *Eng.*, vol. 4, no. 2, pp. 1071–1115, 2023, doi: 10.3390/eng4020063.
- [2] N. Yaseen, “From Counters to Telemetry: A Survey of Programmable Network-Wide Monitoring,” *Network*, vol. 5, no. 3, art. 38, 2025, doi: 10.3390/network5030038.
- [3] B. Claise, “Cisco Systems NetFlow services export version 9,” RFC 3954, Oct. 2004.
- [4] P. Phaal, S. Panchen, and N. McKee, “InMon Corporation’s sFlow: A Method for Monitoring Traffic in Switched and Routed Networks,” RFC 3176, Sep. 2001.
- [5] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
- [6] N. McKeown *et al.*, “OpenFlow: Enabling innovation in campus networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.
- [7] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: Scaling flow management for high-performance networks,” in *Proc. ACM SIGCOMM*, Toronto, ON, Canada, Aug. 2011, pp. 254–265.
- [8] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, “OpenTM: Traffic matrix estimator for OpenFlow networks,” in *Proc. Passive and Active Measurement Conf. (PAM)*, Zurich, Switzerland, Apr. 2010, pp. 201–210.
- [9] Z. Su, T. Wang, Y. Xia, and M. Hamdi, “FlowCover: Low-cost flow monitoring scheme in software defined networks,” in *Proc. IEEE GLOBECOM*, Austin, TX, USA, Dec. 2014, pp. 1956–1961.
- [10] Z. Su, T. Wang, Y. Xia, and M. Hamdi, “CeMon: A cost-effective flow monitoring system in software defined networks,” *Computer Networks*, vol. 92, pp. 101–115, Dec. 2015.
- [11] G. Liu, S. Guo, B. Xiao, and Y. Yang, “SDN-Based Traffic Matrix Estimation in Data Center Networks Through Large Size Flow Identification,” *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 675–690, 2022, doi: 10.1109/TCC.2019.2944823.
- [12] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, “PayLess: A low cost network monitoring framework for software defined networks,” in *Proc. IEEE/IFIP Network Operations and Management Symp. (NOMS)*, Kraków, Poland, May 2014, pp. 1–9.
- [13] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, “OpenNetMon: Network monitoring in OpenFlow software-defined networks,” in *Proc. IEEE/IFIP Network Operations and Management Symp. (NOMS)*, Kraków, Poland, May 2014, pp. 1–8.
- [14] P. K. Shukla, P. Maheshwary, E. K. Subramanian, V. J. Shilpa, and P. R. K. Varma, “Traffic Flow Monitoring in Software-Defined Network Using Modified Recursive Learning,” *Physical Communication*, vol. 57, art. 101997, 2023, doi: 10.1016/j.phycom.2022.101997.
- [15] D. K. Dake, J. D. Gadze, G. S. Klogo, and H. Nunoo-Mensah, “Traffic Engineering in Software-Defined Networks Using Reinforcement Learning: A Review,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 5, pp. 330–345, 2021.
- [16] Y. He, G. Xiao, J. Zhu, T. Zou, and Y. Liang, “Reinforcement learning-based SDN routing scheme empowered by causality detection and GNN,” *Frontiers in Computational Neuroscience*, vol. 18, art. 1393025, 2024, doi: 10.3389/fncom.2024.1393025.
- [17] L. Li, W. Chu, J. Langford, and R. E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” in *Proc. 19th Int. Conf. World Wide Web (WWW)*, Raleigh, NC, USA, Apr. 2010, pp. 661–670.
- [18] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, “A tutorial on Thompson sampling,” *Found. Trends Mach. Learn.*, vol. 11, no. 1, pp. 1–96, Jul. 2018.
- [19] H. Yahyaoui, M. F. Zhani, O. Bouachir, and M. Aloqaily, “On minimizing flow monitoring costs in large-scale software-defined networks,” *Int. J. Network Management*, p. e2220, 2023.
- [20] N. Richerzhagen, P. Lieser, B. Richerzhagen, B. Koldehofe, I. Stavrakakis, and R. Steinmetz, “Change as chance: Transition-enabled monitoring for dynamic networks and environments,” in *Proc. 14th IEEE/IFIP Wireless On-demand Network Systems and Services Conf. (WONS)*, Isola 2000, France, Feb. 2018, pp. 51–58.
- [21] S. Orlowski, R. Wessälly, M. Pióro, and A. Tomaszewski, “SNDlib 1.0—Survivable Network Design Library,” *Networks*, vol. 55, no. 3, pp. 276–286, 2010, doi: 10.1002/net.20371.
- [22] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, “Providing public intradomain traffic matrices to the research community,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, Jan. 2006, doi: 10.1145/1111322.1111341.
- [23] B. Lantz, B. Heller, and N. McKeown, “A Network in a Laptop: Rapid Prototyping for Software-Defined Networks,” in *Proc. 9th ACM Workshop on Hot Topics in Networks (HotNets-IX)*, 2010.
- [24] R. Kubo, T. Fujita, Y. Agawa, and H. Suzuki, “Ryu SDN Framework—Open-source SDN Platform Software,” *NTT Technical Review*, vol. 12, no. 8, pp. 18–22, Aug. 2014, doi: 10.53829/ntr201408fa4.
- [25] Ryu Project Team, *RYU SDN Framework—English Edition (Ryubook 1.0)*, 2014. Available online: <https://book.ryu-sdn.org/en/html/>