Forecasting Routing Inputs

Jean Mégret
ETH Zürich
Zürich, Switzerland
megretj@ethz.ch

Tibor Schneider
ETH Zürich
Zürich, Switzerland
sctibor@ethz.ch

Laurent Vanbever
ETH Zürich
Zürich, Switzerland
lvanbever@ethz.ch

ABSTRACT

BGP messages exchanged over peering sessions in the Internet are currently believed to be unpredictable. Today, network management systems (e.g., for verification, analysis, and optimisation) manage this unpredictability in one of two ways: They either consider future routing inputs in a case-by-case manner, or they consider all possible ones, including highly unlikely scenarios. Thus, the impact or risk of routing inputs on the network is either under- or overestimated.

In this paper, we present a probabilistic perspective to bridge the gap between these two extremes. We explore the idea of building forecasting models of routing inputs and begin to answer the questions of how feasible it is and what it is useful for. This view paves the way for a large class of new systems trying to provide guarantees in spite of uncertainty.

CCS CONCEPTS

• **Networks** → *Network performance analysis*; **Network manageability**; **Network reliability**.

KEYWORDS

Probabilistic Modelling, Routing Inputs, Verification

ACM Reference Format:
Jean Mégret, Tibor Schneider, and Laurent Vanbever. 2026. Forecasting Routing Inputs. In *The 25th ACM Workshop on Hot Topics in Networks (HotNets '26), November 16-17, 2026, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3847290.3859950

1 INTRODUCTION

Networks operate in uncertain environments: traffic fluctuates, devices fail, (external) routes appear and disappear. Any of these changes can affect service quality, sometimes drastically. An important aspect of an operator's job is therefore to reason about and manage this uncertainty by making sure that their network performs well *the majority of the time.*

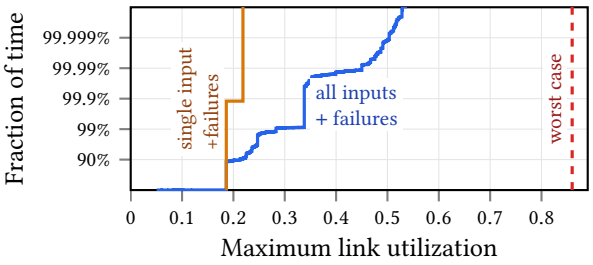


Figure 1: Only considering current routing inputs (like the majority of related work) leads to significantly underestimating risk of congestion, while finding the worst-case state (like Velo [22]) overestimates it.

To reason about uncertainty, operators typically evaluate their network behaviours under likely disruptions/changes. For instance, to reason about traffic uncertainty, operators evaluate the ability of their network to sustain progressively larger traffic demands [7]. Similarly, to reason about failure uncertainty, operators evaluate the ability of their network to accommodate increasingly unlikely failure scenarios, e.g. considering all single-link failures, before all two-link failures, etc. [4, 12, 21] or by relying upon probabilistic failure models [3, 10, 11, 13, 25, 29].

For networks carrying mostly internal traffic, such as data centers, reasoning about the most likely traffic fluctuations and failures is (often) enough to capture the main sources of uncertainty. For ISPs and many distributed WANs, this is not the case. Changes in external BGP routes can significantly affect the network state and performance [22, 28].

Unfortunately, no technique exists to reason about the likelihood of external routing inputs. Instead, existing systems either reason about the *current* external routing state [25, 26] or the *worst-case* one, however unlikely [22]. For many operational questions though, both approaches fail to provide meaningful answers.

As an illustration, consider an ISP operator tasked with assessing whether the network has sufficient capacity or requires further provisioning. Figure 1 shows how the Maximum Link Utilization (MLU) evolves in their network considering: *(i)* failures and fixed (most likely) routing inputs (left curve); *(ii)* failures and changing routing inputs (middle curve); and *(iii)* worst-case failures and routing inputs (right

This work is licensed under a Creative Commons Attribution 4.0 International License.
HotNets '26, November 16-17, 2026, Salt Lake City, UT, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3029-0/2026/11
https://doi.org/10.1145/3847290.3859950

curve).[1] If the operator were to consider the impact of failures on a single routing input, they would likely not consider upgrading their infrastructure. In contrast, if the operator were to consider failures and the worst-case routing inputs (as computed by [22]), the MLU is worryingly close to 1, strongly implying the need for increasing the capacity. Both these decisions would be misguided though, as the actually observed utilization lies between the two extremes.

Capacity planning is only one of several operational tasks that would benefit from a probabilistic model of routing inputs. For instance, such a model could reveal that small routing changes are likely to cause traffic entering at one edge of the network to traverse the entire backbone before exiting, motivating different traffic-engineering policies or even peering strategy. More generally, it would allow operators to quantify the risk of service disruption caused by external routing, identify the routing scenarios contributing most to that risk, and establish probabilistic Service Level Agreement (SLA) guarantees.

Yet, to the best of our knowledge, no probabilistic model for routing inputs exists. [25] even claims: *"while learning a precise internal failure model is possible, [...] this is not the case for external announcements."*

Deriving a probabilistic model for routing inputs is indeed much more challenging than for failures or traffic changes. In fact, the difference is similar to how predicting the weather is more challenging than predicting the outcome of coin tosses:

Large state space. Like meteorological variables, the space of routing input is huge: one needs to consider all possible routes, to all destinations, across all (BGP) sessions. Larger state spaces require more evidence to base a model on, leading to fundamentally more inaccurate models.

No stationary distribution. The outcome of a coin toss has a stationary distribution, which is revealed by repeated tossing. In contrast, the probability of it being sunny tomorrow depends on when it is asked: the distribution *changes* over time. Correspondingly, the probability of seeing different routing inputs also changes over time. For instance, any peering (dis-)appearing can change the distribution of routing messages observed in remote networks.

While there is no denying that predicting routing inputs is hard, we do not think it is impossible. Indeed, our analysis reveals that distributions of routing inputs are heavily skewed and sparse. Skewness is an opportunity: with few well selected routing inputs one can effectively cover most of what is likely to happen. On the flip-side, sparsity raises generalization issues, especially for predicting unforeseen inputs, fundamentally leading to inaccuracies in trained models. Yet the juice is worth the squeeze, after all, entire industries rely on weather forecasts daily [20]; we believe networks operations should do the same with routing inputs.

In the rest of the paper, we shed light on the numerous opportunities that a forecasting model of routing inputs would enable for network operators. We formalize the design space for building such models, considering training and inference. We further demonstrate their feasibility by constructing a simple frequency-based probabilistic model which already achieves an accuracy of up to 90% in forecasting routing inputs on a 24-hour horizon, which is already useful for many operational tasks.

We also describe a promising research agenda towards developing highly accurate models which would enable validating high-availability SLAs (e.g., 4 or 5 "nines"). To do so, we suggest incorporating domain specific knowledge to design parametric models that describe routing changes, and learn these parameters from historical data. We also discuss the potential in employing recent machine learning techniques, such as Transformer-based models [5, 14], and what limits they could hit.

Overall, we believe that this paper paves the way towards more useable and holistic network operation tools by using probabilistic models of routing inputs and closing a more complete view of network environments.

2 MOTIVATION

The following demonstrates the usefulness of forecasting routing inputs in a variety of network operations tasks and summarises them with operational questions.

Probability of an outcome. Operators often make strategic decisions to improve network behavior, like which peering agreements to pursue or which links to add or upgrade. Say operators observe most traffic currently leaves through the east router and suspect it could congest that egress link. Thus, they consider negotiating new peering agreements. But should they do so at the IXP in the north, or at the one in the west? Routing inputs sent today by the IXPs indicate north is better, but is this reliable? A probabilistic model of routing inputs could determine which egress has a lower probability of forwarding traffic to neighbouring networks, justify prioritising new peerings for that egress.

> **Q1**: Does a property hold with probability $\geq q$?
>
> **Q2**: What is the probability of a property to hold?

These two questions may appear to be similar, but they pose different requirements on a probabilistic model. **Q1** does not need to find the exact probability, and a model may therefore be pessimistic. For instance, if $q = 0.5$, it suffices to find a set X of routing inputs in which the property holds that collectively appear with probability $\geq 50\%$. However, answering **Q2** requires finding the exact probability.

[1] The plot is based on real traffic matrices, real routing inputs observed at an IXP, and simulated failures according to an established model [25].

Verifying SLAs and computing risk. Networks commonly provide service-level agreements (SLAs) to customers, such as guaranteeing service availability of $\geq 99.99\%$. In contrast to **Q1**, service availability is not just a true or false property; instead it is measured by the severity of an outage: An event that degrades a single service has a lower cost than one that stops all services from functioning. If operators specify a cost, the expected risk can be computed by multiplying each event's probability with its cost.

This notion of risk can be extended to other properties as well. One example is to estimate the risk of the network being congested. Operators could do so by expressing the cost as the MLU and performing an experiment similar to Figure 1.

> **Q3**: Is the expected risk of a property $\leq r$?

Finding the most pressing issues. If operators deem the risk too high, and want to mitigate it, they must know the culprit: a set of inputs that are both likely and cause significant disruption. Thus, operators should be able to find the input with the highest associated risk, or iterate over all of them in decreasing order of risk. Say a probabilistic model finds that the majority of traffic leaving on the east router is both likely and causes significant congestion. Operators could then take targeted action in order to reach their availability targets.

This is not limited to risk analysis, though. Classical network verifiers either prove the specification, return a single counter example, or fail to find an answer after the given time budget. While the first outcome is insightful, the other two are not: A counter example may describe *any* input in which the property is violated, regardless of how likely that input is. Likewise, if the verifier times out, its computation is wasted without producing any meaningful output. With a probabilistic model of routing inputs, we could go one step further: We could return the most likely counter example, that is, the one operators should focus on first, or, on a timeout, report the probabilistic space that was successfully explored and is proven to satisfy the property.

> **Q4** What is the most likely routing input in which a property is violated, or the most risky one?
>
> **Q5** Can we iterate over all relevant routing inputs (e.g., counter examples) in decreasing order of likelihood?

Finally, note that the accuracy of the model has a downstream effect on the guarantees it can provide. For instance, if the accepted service disruption risk from **Q1** is $1 - q = 0.01\%$, i.e., four nines of SLA guarantees, a model with $\geq 99.99\%$ accuracy is required to guarantee that the SLA is enforced. On the other hand, ordering events relative to their probability does not require such a precise model. Especially as, even if a model would mistakenly return a near-optimal result, addressing it would still improve the network significantly, especially when compared to the status quo, in which an unlikely (or even impossible) one may be returned.

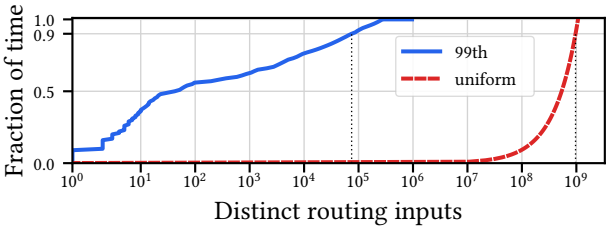


Figure 2: For 99% of the analysed prefixes (orange), the number of likely routing inputs is tiny compared to the theoretical space of possible routing inputs (purple).

Skewness and sparsity. All these questions require a probabilistic model for routing inputs, although with different requirements, that is, capabilities and accuracies. Yet, is it even *possible* to build a forecasting model for routing inputs?

As previous work recognised [25], building a model for routing inputs is extremely challenging. This can be attributed to the vast and exponential space of possible messages and their non-stationarity. Yet, we find that actual distributions of routing inputs are extremely **skewed** and **sparse**. Figure 2 counts the number of distinct routing inputs (states) required to cover a certain fraction of a month worth of real routing inputs on 30 sessions. For 99% of all analysed prefixes (orange line) less than ten thousand states are needed to cover 90% of the month. In contrast, even if we were to simplify the problem and only consider which set of neighbour advertises the best route and say that states were equally likely (purple line), one would need to consider over four orders of magnitude more states for the same confidence.

Sparsity poses challenges for generalisation, especially for predicting unseen states. Despite that, skewness often prevails as even a simple model presented in section 3.5 can already predict future routing inputs with up to 90% accuracy.

3 MODELLING ROUTING INPUTS

In this section, we propose concrete ways of using (section 3.1) and building (3.2) models of routing inputs. We identify challenging modelling dimensions (3.2), propose a formalization (3.3) and describe how to use models despite imperfections (3.4). Finally, to put this into perspective, we present an initial model based on real data (3.5).

3.1 Using distributions over routing inputs

A model generating distributions over network environments is the right tool to answer the questions in section 2. We convey the intuition of why by framing our problem in a

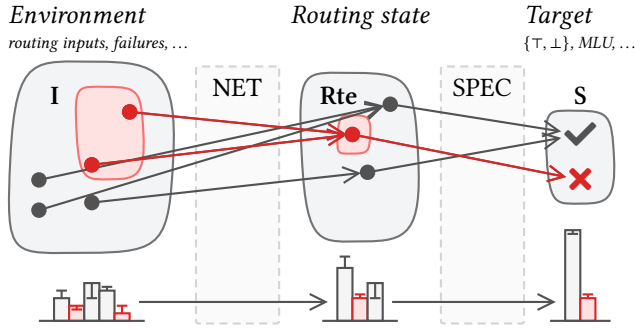


Figure 3: Functional depiction of network verification and analysis. A distribution (shown as bar plots) can be pushed through the functions towards spaces of interest. The total variation (shown as "error bars") between two distributions can only become smaller.

general functional view of network verification and analysis depicted in Figure 3, analogous to the one presented in [23]. A network and its configuration form a function NET from the environments to routing states **Rte**; NET is the *execution* of a particular environment. The specification is a function SPEC from the routing states **Rte** to some target space **S**; SPEC is an *analysis* of a routing state. For example, computing the normalized MLU, is a function SPEC : **Rte** $\to [0, 1]$. Also, any flow property ϕ, such as network-wide reachability, naturally corresponds to: SPEC : **Rte** $\to$ **Bool**, $x \mapsto x \models \phi$.

It might not be directly clear how a distribution over the environment $\mathcal{D}_{\mathbf{I}}$ can answer, for instance, questions **Q1**. What we need is rather a distribution over the target space, $\mathcal{D}_{\mathbf{S}}$. Then we can say what the probability of an element of **S** is, e.g. reachability is true with probability 0.8 and false with 0.2. Fortunately, we can get $\mathcal{D}_{\mathbf{S}}$ by "pushing" $\mathcal{D}_{\mathbf{I}}$ first through NET to give us $\mathcal{D}_{\mathbf{Rte}}$, a distribution over **Rte**, and then through SPEC to get $\mathcal{D}_{\mathbf{S}}$. Furthermore, Figure 3 makes clear how bounds on the error of $\mathcal{M}_{\mathbf{I}}$ is squeezed onto $\mathcal{M}_{\mathbf{Rte}}$ and $\mathcal{M}_{\mathbf{S}}$. Finally, we can compactly and unambiguously describe the high level questions, the first step towards developing algorithms to solve them. **Q1** asks to find a set $X \subseteq \mathbf{I}$ such that $\left(\text{SPEC}(\text{NET}(x)) = \top, \forall x \in X\right)$ and $\left(\mathbb{P}_{\mathbf{I}}[X] \geq q\right)$.

3.2 Designing models

The architectural choices of a model of routing inputs not only influence the performance of the evaluation metric but also move between model complexity and generality. Concretely, we discuss the following questions:

3.2.1 What information about the network configuration should be embedded into the model ?
3.2.2 How should the dependencies between the variables be taken into account?
3.2.3 How to incorporate data with expert knowledge?
3.2.4 What should we expect as the output of our model?
3.2.5 How do we compare and evaluate models?

We ground our analysis of these questions in a real world experiment, shown in Figure 4. As in Figure 2, experiments are randomised across many BGP session sets, prefix sets and start dates. The 99th percentile CDF of the number of routing inputs to cover a time interval is plotted. Each plot modulates a single variable from a base scenario with 1 prefix, 10 sessions, discriminating routes based on AS-path and communities and spanning over 30 days.

It is also useful to reason about the size of the modelling space by considering the total number of different routing inputs that one should consider in general. Considering routes can be described by x bits, our network has ℓ BGP sessions and we track p prefixes, the support of the model is on the order of: $(2^x)^{\ell \cdot p} \geq 2^{\ell \cdot p}$ (1). Reducing the support of distributions is critical to make models more tractable (lower dimensional) and precise (less space for error).

3.2.1 Configuration equivalence classes. Though external announcements are independent of the internal network configuration, the structure of routing input models should depend on equivalence classes (a notion well studied in network verification, e.g. [12]) over the attributes.

This can reduce x in eq. (1) drastically: we may often remove superfluous route attributes from the model. For instance, only a small subset of all possible communities will ever make a difference: those included in the configuration (well-known communities and those set by the operator). Further, only few of the 2^{32} AS numbers are assigned, and the path length is typically bounded to a small number ([19] reports 3.9 on average and it is recommended to filter out long AS Path (ASP) [9]). Hence, based on a configuration, some part of the attributes will be seen as equivalent. In Figure 3, this is characterized by several environments being mapped to the same routing state. Figure 4**(a)** shows that considering larger equivalence classes decreases the support of the distribution: discriminating routing inputs only on the set of sessions with shortest AS path (Dominant) creates larger equivalence classes than considering these sets together with a path length (ASPL). More equivalence classes are created as we start to differentiate routing inputs on the ASP (+Path) and community values (+Comm).

But equivalence classes based on the network configuration can also reduce the exponent in eq. (1). An approach similar to [21] would perfectly split the space of routing inputs on deciding attributes across sessions. Another approach consists in considering the dependencies between sessions and prefixes, as discussed in section 3.2.2.

This surfaces the trade off between generality and specificity. Distinguishing on every single attribute makes the model applicable to any configuration. However, the support

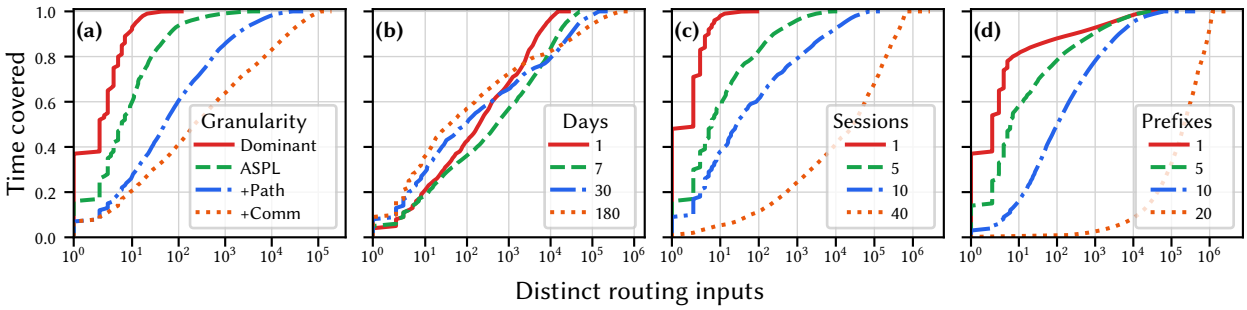


Figure 4: The impact on the size of distribution supports is minor for (a) the granularity of attributes modelled or (b) the time interval. However, increasing the number of (c) sessions or (d) prefixes increases it drastically.

of the distribution will be huge, inevitably increasing the error. On the other hand, finding equivalences reduces the support but it is not cheap to compute and the corresponding models are sensitive to modifications of the configuration, meaning they need to be updated more regularly.

3.2.2 Independence between random variables. In general, sessions and prefixes are dependent. For example, a route being withdrawn at one session will likely also get withdrawn on another session. Also two disjoint prefixes originating from Cameroon might share some critical transatlantic links to reach Brazil. However, directly modelling the joint distribution leads to an exponential explosion in the sample space (eq. (1)) instead of a polynomial increase in the independent case. This is confirmed by Figure 4**(c)** and **(d)** where increasing the number of sessions, resp. prefixes, makes the distributions exponentially broader. Thankfully, to reduce the number of variables, dependent variables can often be correlated in a deterministic way. For example, if routes from several peers are aggregated at edge routers, one may only consider one session per edge router. When the correlation is not so clear, methods such as Bayesian Networks [15], can be used to find dependencies amongst the random variables (r.v.s).

3.2.3 Data-based, parametric or both. Collecting high quality BGP streams is a well studied task [1, 24, 27], so it is safe to assume access to a history of messages, motivating the use of data-driven methods. Furthermore, Figure 4**(b)** shows that the number of days to predict only slightly impacts the number of routing inputs to consider: the skewness and sparseness trends hold across time. Unstructured methods such as simple binning (which we demonstrate in section 3.5), neural networks, or Transformer-based methods [5, 14] are promising to generate distributions.

Akin to Dietmüller et al. [8] Network Traffic Transformers, BGP data could be used to train base models that could later be fine-tuned on specific networks. Then, by averaging multiple sequences of synthetic BGP messages, we can build a distribution over the routing inputs. Transformer architectures are known to generalise well, provided enough training and they can adapt to different initial routing inputs thanks to their attention mechanism. We constructed a simple architecture composed of three small attention-linear blocks surrounded by an embedding and decoding layer. Though the initial results were relatively poor on average, they showed promising TV values for sessions with many BGP messages. We believe that with more training, more parameters and a bigger context, they could learn more general session behaviours.

In general, unless special care is taken, the freedom given by unstructured methods might also hit some limitations. As outlined in section 1 and demonstrated in section 3.5, one cannot assume a stationary distribution. This means that data-driven models should be updated regularly as old data will get outdated. So despite large amounts of historical data, the amount of recent, valid data is potentially scarce. This might become an issue as a large hypothesis space is more prone to over-fitting under scarce data [16].

Alternatively, parametric models constrain the hypothesis space while their values can still be learnt using data. For more robustness, domain knowledge of Internet dynamics accumulated by the community, e.g. [17], should be embedded to the models. Akin to link failures in [25], one could explicitly define the distributions of all the r.v.s of routes, e.g. *the AS path lengths on each session are i.i.d. Poisson distributed with parameters λ, the community i is present with distribution Bernoulli(p_i), etc.*... Or, since BGP messages originate from some event (distant link failure, new peering agreements, ...) one could develop a *local view of the Internet* and model the probability of these distant events happening rather than their downstream effect on the received routes.

3.2.4 Inference type / Interface. Depending on the type of events (i.e., property) under investigation, we might require less from our model than access to the probability of single samples and all marginal probabilities. For example, looking back to **Q1**, a simpler model that, given a $x \in [0, 1]$, returns

a set of routing inputs S that satisfies $\mathbb{P}[S] > x$ can help answer the question in the positive by cranking up x. By simplifying the interface, we leave less space for error, therefore increasing the accuracy of the model.

3.2.5 Validating models. Having a way to describe probabilistic models and build distributions is not enough, it is essential to reason about how well they describe reality. However, there are two fundamental limitations to validating a model of routing inputs: we only have access to a single future and there is also no clear notion of closeness between two routing inputs. The former means there could have been many plausible evolutions of the routing inputs but only one realises itself. The latter restricts the prediction to a binary outcome: what is predicted is either true or false, we cannot directly say the prediction is close.

3.3 Formalising models

We now present a formal model of the probabilistic space of routing inputs with enough expressivity to cover the tradeoffs mentioned previously. Ultimately, our goal is to construct distributions over routing inputs. There are many ways to do this, but we follow a frequentist approach. The probability of a routing input R happening in a time interval t is defined by the proportion of time R is present in t.

3.3.1 Routing inputs as random variables. Let $\mathbf{I}$ be the set of all routing inputs. A routing input $R = \bigcup_{d \in D} R^d$ is composed of, for each destination d in D, routes $R^d = \{r_0, \dots, r_{\ell-1}\}$ where each of the ℓ sessions s_i announce a route r_i. Each r_i contains some attributes or shows no route. For example, focusing here only on two important BGP attributes: the ASP and the set of community values, we characterise r_i via either a list of AS numbers and a set of community numbers or with no route. Other attributes can be supported in the form of special communities, or by introducing more r.v.s.

3.3.2 Towards distributions. We construct a distribution using the following process. A time interval t starts with a routing input R_0. Over t, we will see a sequence of routing messages $m_0, m_1, \dots$ which implies a sequence of routing inputs, generating a set $\mathcal{R}_t = \{R_0, \dots, R_{r-1}\}$. Counting the time τ_i spent in each R_i, we ascribe the probability τ_i/t to each R_i. This defines a Probability Mass Function (pmf) from $\mathbf{I}$ to $[0, 1]$ and leads to the following probability space.

3.3.3 Probability space. A probability space allows us to speak about the occurrence probability of a set of routing inputs with some common property over the model's r.v.s. It is made up of a triple $(\Omega, \mathcal{F}, \mathbb{P})$ (*e.g.* , [6]). Ω is the sample space, which describes all possible outcomes of a single execution of a model, *i.e.* a sample of the r.v.s. In our case, it is a routing input: $R \in \mathbf{I}$, which is a sample of the r.v.s representing route attributes at each session. $\mathcal{F}$ is the event space containing all subsets of Ω. This is the interface of the model (section 3.2.4) and represents a set of routing inputs. Finally, $\mathbb{P} : \mathcal{F} \to [0, 1]$ is the probability measure. Given this, a pmf (or distribution) maps each event to a probability.

3.4 Evaluating models

The "single future" problem described in section 3.2.5 leads us to abstract the model from the data it is trained on, making the training data part of the input. Then, to evaluate a model architecture, we record its performance on many scenarios (different time ranges, prefix- and sessions-sets) by comparing the predicted distribution and the true distribution.

There are many ways (*e.g.* [18]) to compare the model distribution $\mathcal{M}$ to the true distribution $\mathcal{D}$ but only few are revealing in terms of how they can be used for providing guarantees to the questions presented in section 2. As hinted in section 3.2.5, since there is no direct notion of closeness between samples we propose to look at the problem from a classification perspective. But, beyond simply counting the fraction of samples in each class (true/false-positive/negative), we take into account their probability. The Total Variation (TV) distance over a space X is a symmetric measure of how much probability mass is misclassified.

$$\mathrm{TV}_X(\mathcal{D}_X, \mathcal{M}_X) = \frac{1}{2} \sum_{i \in X} |\mathcal{D}_X[i] - \mathcal{M}_X[i]|$$

One can further specify for which class the variation should be computed on. For instance, the false negative variation $\mathrm{TV}^{FN} = \mathrm{TV} - \mathrm{TV}^{TP}$ tells us how much of the true distribution our prediction "misses". To bound the error on individual samples, ϵ measures how far off the probability of any single routing input is.

$$\mathrm{TV}^{FN}_X(\mathcal{D}, \mathcal{M}) = \sum_{i \in \mathrm{FN}} \mathcal{D}[i], \quad \epsilon_X(\mathcal{D}, \mathcal{M}) = \max_{i \in X} |\mathcal{D}[i] - \mathcal{M}[i]|$$

3.4.1 Usefulness of imperfect models. Reality obliges us to rely on models with errors to get confidence in answering operational questions. For this, we again tap into Figure 3 and see that the variation between $\mathcal{M}_\mathbf{I}$ and $\mathcal{D}_\mathbf{I}$ is an upper bound on the variation at the **Rte** and S level. Using the fact that functions are univalent, we can show that:

Proposition 3.1. $TV_\mathbf{I} \geq TV_\mathbf{Rte} \geq TV_\mathbf{S}$

Proof. By definition in section 3.1, $\forall i \in \mathbf{I}$:

$$\mathcal{M}_\mathbf{I}[i] = \mathcal{M}_\mathbf{Rte}[\mathrm{NET}(i)] = \mathcal{M}_\mathbf{S}[\mathrm{SPEC}(\mathrm{NET}(i))] \tag{2}$$

The same holds for $\mathcal{D}$. Now, since functions are univalent relations, i.e. a function is a many-to-one relation, mapping every point of the domain $\mathbf{I}$ might yield the same codomain

Rte points multiple times. Hence:

$$\begin{aligned} 2\,\mathrm{TV}_{\mathbf{I}} &= \sum_{i\in\mathbf{I}} |\mathcal{D}_{\mathbf{I}}[i] - \mathcal{M}_{\mathbf{I}}[i]| \\ &\overset{\text{eq. (2)}}{=} \sum_{i\in\mathbf{I}} |\mathcal{D}_{\mathbf{Rte}}[\mathrm{NET}(i)] - \mathcal{M}_{\mathbf{Rte}}[\mathrm{NET}(i)]| \\ &\geq \sum_{r\in\mathbf{Rte}} |\mathcal{D}_{\mathbf{Rte}}[r] - \mathcal{M}_{\mathbf{Rte}}[r]| = 2\,\mathrm{TV}_{\mathbf{Rte}} \end{aligned}$$

And the same holds for $\mathrm{TV}_{\mathbf{Rte}} \geq \mathrm{TV}_{\mathbf{S}}$. □

Proposition 3.1 helps answer questions from section 2 with an imperfect $\mathcal{M}_{\mathbf{I}}$! This is capital, because it lets us estimate whether a model is sufficiently precise to be useful. These properties however are to be taken with a grain of salt: the TV and ϵ are only statistical measures. In **Q1**, $\mathbb{P}_{\mathbf{S}}[\phi] \geq q$ holds if we find a *sufficient certificate*: a set of routing inputs that all satisfy the property ϕ and such that the probability mass of this set is greater than $q + \mathrm{TV}_I$. The intuition behind this is that the probability given by our model on a subset of routing inputs is upper-bounded by TV_I.[2] For instance, for the model presented in Figure 5, the lowest TV_I achieved is around 0.1, meaning that we can verify properties up to $1 - \mathrm{TV}_I = 90\%$. If the property is satisfied on the whole model support, we can provide an even tighter bound: $\mathbb{P}_{\mathbf{S}}[\phi] \geq q$ holds if $1 - \mathrm{TV}_I^{FN} \geq q$. For **Q2**, $\mathbb{P}[\phi = \top]$ will be at most TV_I off. Computing the risk in **Q3** depends on the **S**. For example, if $\mathbf{S} = [0, x]$ (e.g. unnormalized MLU), then $\mathbb{E}(\mathcal{M}_{\mathbf{S}})$ is off by at most $x \cdot \mathrm{TV}_{\mathbf{I}}$. Finally, in **Q4-5**, though the exact order of the individual routing inputs might be shuffled around by model imprecisions, we know that two environments more than $2\epsilon(\mathcal{M}_{\mathbf{I}})$ apart are placed in the correct relative order.

3.5 An initial model

To provide a baseline for more elaborate models and illustrate the previous sections, we directly apply the description from section 3.3.2, counting the proportion of time spent in each routing input within a history t_h, to construct a model.

Figure 5 depicts the TV and ϵ from section 3.4 between the predicted distribution "trained" over t_h and the true distribution constructed over the consecutive t_f days. Results are randomised over many start dates, unique prefixes and sets of 10 sessions. We observe that the prediction is best for short prediction horizons. This seems to imply that routing inputs undergo "irreversible" changes. Finally we notice that, regardless of the training days, the prediction gets worse with more data, suggesting that the distribution is not stationary. More history only biases the distribution towards outdated routing inputs. These observations expose the main flaw of this model: over-fitting. Worse, this type of model is not generative: any combination which has never been seen historically will be assigned zero probability.

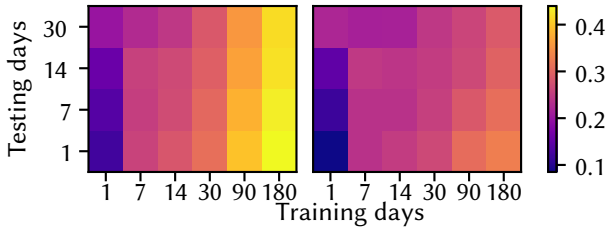


Figure 5: As expected, the average TV (left) > ϵ (right). For short intervals, the TV stays around 0.1.

4 FUTURE WORK

We acknowledge the limitation of using an average TV over multiple scenarios as a way to evaluate a model in section 3.4. Different scenarios might be statistically dependent and the total number of scenarios remains small. Despite this being a good proxy, we do not yet make claims on *strict* guarantees. Future work should focus on strengthening this with methods inspired from resampling methods (*e.g.* bootstrapping) or conformal prediction [2] to increase the confidence of statistics beyond empirical averages. Creating embeddings of routing inputs can help assign a notion of closeness and evaluate predictions more naturally.

Regardless of the guarantees, observations from section 3.5, suggest more sophisticated models should be built to reach SLA-level confidence. From the discussion in section 3.2, we believe this can be done using a combination of domain knowledge and learning methods. Especially, section 3.2.3 suggests several models, including a Transformer-based architecture. Also, our discussion in section 3.2.1 about simplifying the model depending on the configuration should lead up to a tool that automatically generates the granularity given the configuration files.

Finally, a model will really begin to shine once packaged into a complete verification system composed of a simulator, a data-collection framework, pre-configured model architectures and standardised specifications.

Ethical Concerns. This work raises no ethical concerns.

Acknowledgements. This project was supported by an ERC Starting Grant (SyNET) No. 851809.

[2] Formally, we can state it in the following way: $\mathbb{P}_{\mathbf{S}}[\phi] \geq q$ holds if we find a set of routing inputs X such that $\forall x \in X : \mathrm{NET}(x) \models \phi$, and $(\mathcal{M}_{\mathbf{I}}[X] \geq q + \mathrm{TV}_I)$.

proof: $\mathbb{P}_{\mathbf{S}}[\phi] = \mathcal{D}_{\mathbf{S}}[\Phi_{\mathbf{S}}] \geq \mathcal{M}_{\mathbf{S}}[\Phi_{\mathbf{S}}] - \mathrm{TV}_{\mathbf{S}} \overset{3.1}{\geq} \mathcal{M}_{\mathbf{S}}[\Phi_{\mathbf{S}}] - \mathrm{TV}_{\mathbf{I}} \overset{\text{eq. (2)}}{=} \mathcal{M}_{\mathbf{I}}[\Phi_{\mathbf{I}}] - \mathrm{TV}_{\mathbf{I}} \geq \mathcal{M}_{\mathbf{I}}[X] - \mathrm{TV}_{\mathbf{I}} \geq q$.
If the last inequality holds, the whole chain holds. □

REFERENCES

[1] Thomas Alfroy, Thomas Holterbach, Thomas Krenc, K. C. Claffy, and Cristel Pelsser. 2024. The Next Generation of BGP Data Collection Platforms. In *Proceedings of the ACM SIGCOMM 2024 Conference* (Sydney, NSW, Australia) *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 794–812. https://doi.org/10.1145/3651890.3672251

[2] Anastasios N. Angelopoulos and Stephen Bates. 2021. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. *CoRR* abs/2107.07511 (2021). arXiv:2107.07511 https://arxiv.org/abs/2107.07511

[3] Behnaz Arzani, Sina Taheri, Pooria Namyar, Ryan Beckett, Siva Kesava Reddy Kakarla, and Elnaz Jalilipour. 2025. Raha: A General Tool to Analyze WAN Degradation. In *Proceedings of the ACM SIGCOMM 2025 Conference* (São Francisco Convent, Coimbra, Portugal) *(SIGCOMM '25)*. Association for Computing Machinery, New York, NY, USA, 184–202. https://doi.org/10.1145/3718958.3754348

[4] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2017. Minesweeper: A General Approach to Network Configuration Verification. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 155–168. https://doi.org/10.1145/3098822.3098834

[5] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. 2021. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 27381–27394. https://proceedings.neurips.cc/paper_files/paper/2021/file/e614f646836aaed9f89ce58e837e2310-Paper.pdf

[6] Patrick Billingsley. 2012. *Probability and measure* (anniversary ed. ed.). Wiley, Hoboken, N.J. https://toc.library.ethz.ch/objects/pdf/z01_978-1-118-12237-2_01.pdf

[7] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjørner, Asaf Valadarsky, and Michael Schapira. 2019. TEAVAR: striking the right utilization-availability balance in WAN traffic engineering. In *Proceedings of the 2019 ACM SIGCOMM Conference*. 29–43.

[8] Alexander Dietmüller, Siddhant Ray, Romain Jacob, and Laurent Vanbever. 2022. A New Hope for Network Model Generalization. In *HotNets '22: Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. Association for Computing Machinery, Austin, TX, USA. https://doi.org/10.1145/3563766.3564104

[9] Jerome Durand, Ivan Pepelnjak, and Gert Döring. 2015. BGP Operations and Security. RFC 7454. https://doi.org/10.17487/RFC7454

[10] Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. 2016. Probabilistic NetKAT. In *Programming Languages and Systems*, Peter Thiemann (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 282–309.

[11] Timon Gehr, Sasa Misailovic, Petar Tsankov, Laurent Vanbever, Pascal Wiesmann, and Martin Vechev. 2018. Bayonet: probabilistic inference for networks. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, Philadelphia PA USA, 586–602. https://doi.org/10.1145/3192366.3192400

[12] Aaron Gember-Jacobson, Raajay Viswanathan, Aditya Akella, and Ratul Mahajan. 2016. ARC: Fast Control Plane Analysis Using an Abstract Representation. In *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, Florianopolis Brazil, 300–313. https://doi.org/10.1145/2934872.2934876

[13] Nick Giannarakis, Alexandra Silva, and David Walker. 2021. ProbNV: probabilistic verification of network control planes. *Proceedings of the ACM on Programming Languages* 5, ICFP (Aug. 2021), 1–30. https://doi.org/10.1145/3473595

[14] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. 2022. Biological sequence design with gflownets. In *International conference on machine learning*. PMLR, 9786–9801.

[15] Srikanth Kandula, Dina Katabi, and Jean-Philippe Vasseur. 2005. Shrink: a tool for failure diagnosis in IP networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data (MineNet '05)*. Association for Computing Machinery, New York, NY, USA, 173–178. https://doi.org/10.1145/1080173.1080178

[16] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. 2021. Physics-informed machine learning. *Nature Reviews Physics* 3, 6 (2021), 422–440.

[17] Olaf Maennel and Anja Feldmann. 2002. Realistic BGP traffic for test labs *(SIGCOMM '02)*. Association for Computing Machinery, New York, NY, USA, 31–44. https://doi.org/10.1145/633025.633029

[18] Mike Nguyen. 2025. *Foundations of Data Analysis*. Vol. 1. Springer Cham. https://doi.org/10.1007/978-3-032-01858-8

[19] Potaroo 2026. AS6447 BGP Routing Table Analysis Report. https://bgp.potaroo.net/as6447/. Maintained by Geoff Huston. Accessed 16 April 2026.

[20] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. 2025. Probabilistic weather forecasting with machine learning. *Nature* 637, 8044 (Jan. 2025), 84–90. https://doi.org/10.1038/s41586-024-08252-9 Publisher: Nature Publishing Group.

[21] Tibor Schneider, Jean Mégret, and Laurent Vanbever. 2025. Guided Exploration of Control-Plane Routing States. In *Proceedings of the 33rd IEEE International Conference on Network Protcols*.

[22] Tibor Schneider, Stefano Vissicchio, and Laurent Vanbever. 2025. Verifying maximum link loads in a changing world. In *Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation*.

[23] Tibor Jan Schneider. 2025. *Extending the Limits of Formal Methods in Network Operations*. Ph. D. Dissertation. ETH Zurich. https://doi.org/10.3929/ETHZ-C-000790286

[24] John Scudder, Rex Fernando, and Stephen Stuart. 2016. BGP Monitoring Protocol (BMP). RFC 7854. https://doi.org/10.17487/RFC7854

[25] Samuel Steffen, Timon Gehr, Petar Tsankov, Laurent Vanbever, and Martin Vechev. 2020. Probabilistic Verification of Network Configurations. In *Proceedings of the 2020 ACM SIGCOMM Conference*. 750–764.

[26] Kausik Subramanian, Anubhavnidhi Abhashkumar, Loris D'Antoni, and Aditya Akella. 2020. QARC: Detecting network load violations for distributed control planes. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*. Association for Computing Machinery, 974–988. https://doi.org/10.1145/3385412.3385976

[27] Stefano Vissicchio, Luca Cittadini, Maurizio Pizzonia, Luca Vergantini, Valerio Mezzapesa, and Maria Luisa Papagni. 2010. Beyond the Best: Real-Time Non-Invasive Collection of BGP Messages. *INM/WREN* 12 (2010).

[28] Dan Wang, Peng Zhang, and Aaron Gember-Jacobson. 2024. Expresso: Comprehensively Reasoning About External Routes Using Symbolic Simulation. In *Proceedings of the 2024 ACM SIGCOMM Conference*. Association for Computing Machinery, New York, NY, USA.

[29] Peng Zhang, Dan Wang, and Aaron Gember-Jacobson. 2022. Symbolic router execution. In *Proceedings of the ACM SIGCOMM 2022 Conference*. ACM, Amsterdam Netherlands, 336–349. https://doi.org/10.1145/3544216.3544264