TIPSY: Predicting where traffic will ingress a WAN

Michael Markovitch
Brown University, USA
mmarkovi@cs.brown.edu

Sharad Agarwal
Microsoft, USA
sagarwal@microsoft.com

Rodrigo Fonseca
Microsoft, USA
rofons@microsoft.com

Ryan Beckett
Microsoft, USA
rybeckett@microsoft.com

Chuanji Zhang
Microsoft, USA
chuazhan@microsoft.com

Irena Atov
Microsoft, USA
iratov@microsoft.com

Somesh Chaturmohta
Microsoft, USA
someshch@microsoft.com

ABSTRACT

In addition to consumer workloads, public cloud providers host enterprise workloads such as video conferencing and AI+ML pipelines. Enterprise workloads can, at times, overwhelm the available ingress capacity on individual peering links. Traditional techniques to address this problem in the consumer setting do not always apply here, such as use of CDN caches in eyeball networks.

Ingress congestion events necessitate shifting traffic to other peering links at short timescales. While content providers use such techniques in the egress direction, ingress is inherently a different and more challenging problem. Once a packet leaves an enterprise network, it is subject to opaque routing policies that influence the path to the cloud provider.

We present TIPSY, a statistical-classification-based system for predicting the peering link through which a flow will enter a WAN. TIPSY's predictions are used to safely operate a congestion mitigation system that injects BGP withdrawal messages to redirect traffic away from congested peering links. We train TIPSY on traffic data from the Azure WAN, and we demonstrate 76% accuracy in predicting through which 3 peering links (out of thousands) a flow will enter the network after BGP withdrawals.

CCS CONCEPTS

• **Networks** → *Network reliability*; *Network management*; *Network monitoring*; *Public Internet*; • **Computing methodologies** → *Supervised learning by classification*; *Ensemble methods.*

KEYWORDS

WAN, peering, BGP, statistical classification

ACM Reference Format:
Michael Markovitch, Sharad Agarwal, Rodrigo Fonseca, Ryan Beckett, Chuanji Zhang, Irena Atov, and Somesh Chaturmohta. 2022. TIPSY: Predicting where traffic will ingress a WAN. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3544216.3544234

SIGCOMM '22, August 22–26, 2022, Amsterdam, Netherlands

ACM ISBN 978-1-4503-9420-8/22/08…$15.00
https://doi.org/10.1145/3544216.3544234

1 INTRODUCTION

Large cloud and content providers operate wide-area networks (WANs) with hundreds or thousands of peering links with other networks. Such pervasive connectivity with the rest of the Internet improves the performance of traffic that is exchanged with these WANs. However, widespread peering connectivity also comes with challenges and risks. WAN operators must stay ahead of peering link outages and provision sufficient capacity elsewhere to avoid cascading outages. Moreover, as the traffic profile changes over time, operators have to identify new peering opportunities and decommission others. With more peering, operators face increased costs, management complexity, attack surface area, and number of outages. With less peering, users may face increased latency as their traffic traverses longer Internet paths.

Despite careful planning and provisioning of WAN peering, changing application workloads can at times overwhelm the available capacity in the ingress direction on individual WAN peering links. For instance, onboarding new enterprise customers to the cloud as well as increased demand for video conferencing, document hosting, and video AI+ML workloads from existing customers can result in large data transfers concentrated over individual peering links at times. When congestion happens, it can lead to high packet loss and, as a result, degraded user experience.

Congestion incidents for ingress traffic necessitate quickly shifting some traffic from an overwhelmed peering link to other peering links. Unfortunately, unlike in the egress direction, WAN operators have few mechanisms available to control the path that ingress traffic takes to reach their network. While BGP offers some crude mechanisms to signal preferred ingress points (e.g., the multi-exit discriminator attribute or AS path prepending (*e.g.,* [33]), these mechanisms are both coarse grained and heuristic – they may just be ignored by ASes (Autonomous Systems) along the path.

One way to mitigate ingress congestion in a WAN is through the selective withdrawal of BGP advertisements to peers. Large cloud and content providers rely heavily on BGP anycast to advertise their IP address blocks globally. Withdrawing a prefix from one or more peers will re-route traffic through the Internet to an alternative peering link – one that ideally has spare capacity. However, blindly withdrawing BGP advertisements can lead to chaos; if the shifted traffic moves to other peering links that do not have sufficient spare capacity, then those links may themselves become congested.

To enable congestion mitigation for ingress traffic, in this paper we present TIPSY,[1] a system that predicts where traffic will enter a

[1]Traffic Ingress Prediction SYstem

WAN with high accuracy. Despite not having all the private information needed to build a deterministic model of Internet routing, despite large WANs having thousands of peering links over which a flow can arrive, and despite individual ASes spreading their traffic over hundreds of peering links, we find that it is possible for a large cloud provider to predict how traffic will enter their WAN.

At the heart of TIPSY is a system for statistical classification, which is used to identify the most likely ingress peering links for a given flow. TIPSY handles extensive data collection, has a scalable processing pipeline, and produces output to operators to identify which peering links they need to focus on and why. Note that TIPSY predicts the outcome of provided flows and does not predict what the future traffic matrix will be. The latter is a separate problem that prior works have addressed (§ 7).

TIPSY is integrated with the Azure WAN's congestion mitigation system that injects BGP withdrawals to a specific peer when it detects that a peering link is being overwhelmed by ingress traffic. By predicting the most likely new ingress links, TIPSY ensures that the system will only inject such withdrawal messages when, with high probability, the mitigated traffic will shift to new peering links with sufficient spare capacity to handle the increased traffic volume.

To summarize, our contributions in this paper are:

- We provide an overview of the *traffic ingress prediction* problem for WANs, and motivate its use in addressing ingress peering link congestion through prefix withdrawals.
- We build and evaluate TIPSY, a system that uses statistical classification to solve this prediction problem for the Azure WAN. We demonstrate accuracy in predicting the top three links through which 76% of bytes will enter the WAN after BGP withdrawals.
- Using TIPSY, we identify high-risk events that could result in spillover traffic exceeding the provisioned capacity of other peering links on the Azure WAN.

In addition, we consider outages of peering links as degenerate cases where all prefixes are withdrawn at those links. We use these situations as evaluation points and argue that operators can also use TIPSY in capacity planning to address inevitable peering link outages. We expect TIPSY to be applicable to other large WANs as well. However, it is very likely overkill for small networks with few peering links, as the prediction challenge there is trivial.

We believe this is the first work to address this problem. Predicting where traffic will ingress a large WAN is non-trivial. Once an Internet-bound packet leaves its source, it is subject to a variety of opaque routing policies, connectivity constraints, and traffic engineering strategies in different ASes that influence the path it takes. Detailed information about AS-level policies is typically confidential and not available outside that AS. Given the difficulty of predicting ingress Internet paths, to date researchers have largely focused on the problem of traffic engineering within the WAN or of egress traffic [6, 8, 9, 11, 16, 17, 20, 21, 23, 27, 35, 39, 40].

This work does not raise any ethical issues.

2 BACKGROUND & MOTIVATION

Large cloud and content providers operate networks with wide peering surfaces and significant egress traffic from customer and first-party services. Cloud providers also have significant traffic in the ingress direction, due to onboarding of enterprise customers, large-scale hosted storage services, sensor and video analytics pipelines, among others.

Traffic engineering is constantly required to avoid congested links, manage upgrades and maintenance, and to mitigate adverse events. This is a hard problem given the large number of peers and peering links, the diversity of customers, the constant changing nature of traffic, and the asymmetric nature of the control knobs in both directions. While solutions exists to effectively steer traffic in the egress direction towards an external destination [35, 40], the ingress direction poses a much harder problem [33].

With many possible ingress peering points, cloud providers make heavy use of BGP anycast, advertising internal prefixes on most or all peering links. At each AS hop of an incoming flow, there can be multiple routes to take, and the final routes result from independent decisions of each AS in the path. When there is inbound congestion on a link, the most effective solution for the affected network may be to selectively stop advertising one or more prefixes on the affected peering. However, this may trigger a new set of independent decisions by the ASes on the path, causing the traffic to go to other links, which may, in turn, become themselves congested.

This is a significant problem. In the Azure WAN, for example, in the second semester of 2021, we experienced enough ingress congestion events that resulted in a few hundred BGP prefix withdrawals per week, on average. In some of these, withdrawals from one peering link caused cascading congestion that led to a succession of other withdrawals until traffic was sufficiently dispersed.

Predicting the effect of one or more prefix withdrawals on other links can be an enormous aid to operators, as there are usually multiple alternative actions that they can take, such as simultaneous withdrawals in multiple peering links, withdrawals of alternative prefixes, or, in a longer time frame, guide capacity planning.

Techniques to spread user requests over multiple servers, such as DNS-based load balancing, are used effectively by content providers (e.g. [30]), but are insufficient to fully address ingress congestion for cloud providers. Enterprise workloads include long-lived flows [30] with significant ingress bytes, especially with widespread use of IPSec and VPN tunnels to extend on-prem enterprise networks into the public cloud. Those connections cannot be terminated at static CDN caches deep inside ISP networks, and DNS load balancing does not shift flows already in flight. Hence a solution is needed at the routing layer, while DNS-based solutions can continue to be used complementarily for CDN traffic and new flows.

Cascading Ingress Congestion Example On 04 January 2022, around 21:00, a peering link $I1$ with 400 Gbps capacity with peer AS B in location $L1$ in the US hit 90% utilization in the ingress direction. A simplified topology is shown in Figure 1. To mitigate this congestion, a BGP withdrawal was issued for a /10 anycast prefix at $I1$. Utilization subsequently dropped to about 18%. However, peering link $I2$, also with 400 Gbps capacity with the same peer in the same geographic location, took on some of that load and it then exceeded 80% ingress utilization. Another BGP withdrawal was issued for the same prefix at $I2$. This caused peering links $I3$ and $I4$ with the same peer but with only 100 Gbps capacity each in location $L2$ in the US to hit 97% and 85% utilization. Additional BGP withdrawals

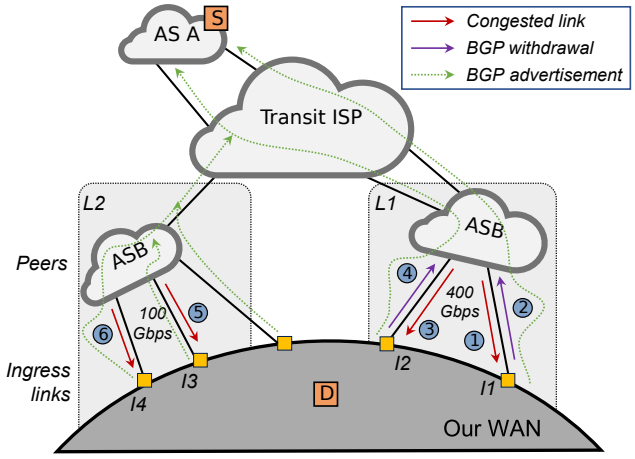

Figure 1: Ingress congestion incident. Green arrows represent BGP advertisements for an example destination D to a source S. Red arrows represent congestion events on ingress links, and purple arrows represent manual BGP withdrawals to relieve link congestion. Blue Numbers indicate the order of events leading to the incident. After withdrawing advertisements from location L1, traffic shifts and congests location L2, which has less peering capacity.

were then issued for that prefix at $I3$ and $I4$, at which point that load was spread diversely enough to not exceed safe utilization.

Ingress congestion events can last for long periods due to time spent detecting congestion from network measurement data and waiting for BGP convergence in one or multiple withdrawal actions. We have observed events lasting many minutes, to hours, to even days (§6). Congestion eventually dissipates due to traffic being shifted to links with sufficient capacity, or the source of the flows reducing traffic load, sometimes in response to poor performance.

TIPSY was not operational at the time of this incident. Predicting the likely outcome of prefix withdrawals in this incident is non trivial. There are hundreds of transit peering connections that the traffic from AS B could have theoretically arrived on. Just focusing on the direct peering connections with AS B still yields 60 different routers in 31 locations across the world. Even if focusing within the same country of US, there are many direct peerings with AS B at 7 different geographic locations.

In post-incident analysis, we trained a TIPSY model on 90 days of data ending on 03 January 2022 and queried it against the flows that arrived on $I1$ to the prefix that was withdrawn. TIPSY correctly identified $I2$ as being at highest risk, getting 50 Gbps additional load, followed by $I3$ and $I4$ getting 47 Gbps of additional load. It also identified 22 additional links, but each expected to receive fewer than 4 Gbps of additional load. Armed with this information, the operator would have had better options. They could have withdrawn the prefix from $I1$, $I2$, $I3$, $I4$ simultaneously at the start of the incident and avoid the cascade of congestion events that resulted in lengthy packet delays and drops. They could have focused on other prefixes that would not have resulted in a cascade and would still shift sufficient traffic to mitigate congestion on $I1$.

This example also demonstrates why this problem cannot be solved by waiting for all ASes to deploy egress traffic engineering (TE). There may not be sufficient additional peering capacity that a neighbor has with the Azure WAN to shift traffic under congestion or outages, but a different neighbor may. Other links that the neighbor has may be less optimal than what a different neighbor has from a path length or latency point of view.

Ingress Traffic Prediction To enable effective ingress traffic engineering, we focus on the problem of predicting the ingress link that traffic will use. Consider a large cloud provider network, such as the Azure WAN, which has thousands of peering links across many different ASes. The Azure WAN connects to some small ASes via a single peering link [5], while larger ASes may have many peering links with the Azure WAN spread over multiple metro regions around the world. Figure 1 shows a high-level depiction of how packets from a source S, in a remote AS A, find a route to server D in the Azure WAN. The ingress path taken from S to D is constrained by the advertisements made by the Azure WAN, but each AS along the path to AS A also makes independent choices as to which path to select and re-advertise. Using BGP anycast, the Azure WAN may advertise the prefix for D on multiple, typically all, peering links. Predicting the ingress link for traffic is a challenging problem for a number of reasons:

(1) Many possibilities: As shown in Figure 1, the combination of BGP anycast with the increasingly densely connected AS topology [7] results in a large diversity of Internet paths that traffic may take from any given source. Moreover, determining the exact path that will be taken would require knowledge of all the inter-domain and intra-domain policies and traffic engineering systems used by each AS along the path(s) from the source to the WAN.

(2) Asymmetry of control: Unlike with egress traffic, where operators can select on which outgoing links to send traffic [35, 40], there is very little that operators can do to steer ingress traffic beyond coarse mechanisms such as BGP MED [18] or selective advertising. Even then, it is hard to predict the outcomes of these measures, as they depend on independent decisions by other ASes.

(3) Lack of visibility: Peering agreements between ASes are not necessarily public, nor always visible through BGP announcements. Even state-of-the-art Internet topology research [22] is based on incomplete data gathered from BGP listeners (and it has limited information about multiple peerings between the same ASes at different locations). Furthermore, BGP advertisements provide information about reachability in a single direction (egress) and there is no guarantee or even likelihood of symmetry.

(4) Constant change: Lastly, the Internet topology is constantly changing, due to intentional changes (peering and de-peering, traffic engineering, etc.), and un-intentional changes (BGP route flapping, fiber cuts, etc.).

To quantify the difficulty of the ingress prediction problem, we analyzed the Internet topology using BGP Monitoring Protocol (BMP) [4] data, and inbound traffic using Internet Protocol Flow Information Export (IPFIX) [3] data, collected from edge routers in the Azure WAN. Details on this data and time window are in §4.1.

We examine the AS topological distance between the Azure WAN and the source of traffic entering our network. We expect that the

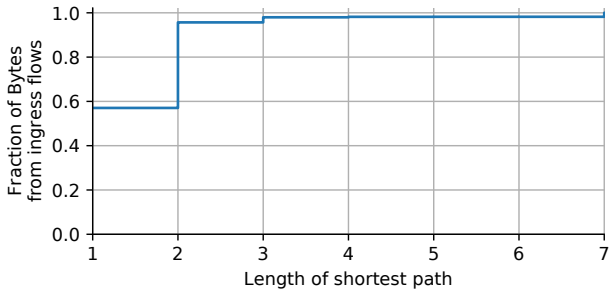

Figure 2: CDF of Bytes by distance of source AS.

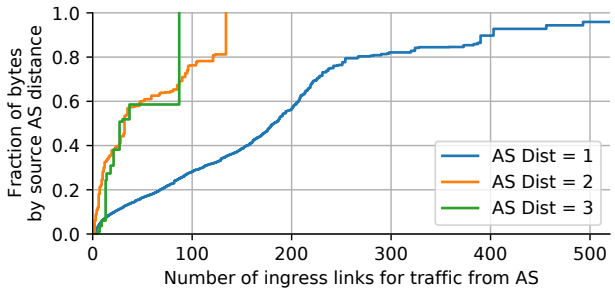

Figure 3: CDF of Bytes from source ASes against the number of our peering links that received it, grouped by AS distance.

further away a source AS is, the more intermediate ASes impose their opaque policies on the route announcements that the Azure WAN makes. Figure 2 shows the CDF of all the bytes that ingressed the Azure WAN during the time period we considered compared to the number of AS hops away the source is [2]. In agreement with past observations that the Internet topology is flattening [7], almost 60% of the bytes come from an AS that peers directly with the Azure WAN, and 98.2% come from ASes at most 3 AS hops away.

We might assume from this finding that the closer a source is to the Azure WAN, the fewer paths the source selects to reach us, as fewer route selection decisions are made. However, Figure 3 shows that this is not the case. For every source AS that sent us traffic, we calculated how many unique peering links their traffic arrived on, and plotted the distribution of ingress bytes across all source ASes, grouped by how many hops away the source AS is. It shows, for example, that 50% of the bytes coming from ASes that are one hop away are sprayed across up to 182 peering links. Many of these peering links are not direct peering links with the neighbor AS, hence it is not accurate to assume that traffic will arrive only on the closest direct peering link. Interestingly, the further away the source is, the fewer peering links will receive their traffic.

We believe part of the reason for this unexpected finding is that large CDNs that we peer with have isolated pockets of their network across the globe that can only reach us through public connectivity, because they lack a global WAN backbone. Additionally, other large ASes that we peer with may employ routing policies to avoid the use of their private long-haul links in favor of public connectivity.

These two characteristics combine to make the problem non-trivial. Despite this difficulty, we next define the problem more precisely, and then demonstrate that simple statistical-classification-based learning can achieve good results in predicting the ingress link for traffic. Later in §5, we show how operators can leverage these learning models to simulate "what-if" scenarios to assist with the challenging task of mitigating ingress congestion.

3 TRAFFIC INGRESS PREDICTION

In this section, we first describe the traffic ingress prediction problem, then frame the problem as a statistical classification problem, and finally define the classification models we will use throughout the rest of the paper.

3.1 Problem definition

The goal of the traffic ingress prediction problem is to predict, for a given traffic flow f, through which (of the possibly thousands) peering links f will enter the network. In this work, we refer to a peering link at the granularity of an individual eBGP session – it may be running on a virtual interface comprised of a bundle of physical links, or it may be running on a virtual interface that is one of many on a physical interface. While we are interested in the ingress peering link, predicting the coarser-grained AS- or PoP-level Internet path [14, 26, 28, 36] is a related, but different problem. Ours is a more feasible problem, and yet still provides substantial value for important WAN operational tasks such as congestion mitigation. We consider flows at the granularity of source and destination prefixes, since BGP operates at that granularity. We define these flow aggregates in §3.2 and §4.2.

We say that a flow f enters a WAN through peering link l, written as $\text{ingress}(f) = l$. The goal of the prediction problem is then, given a set of peering links $\mathcal{L}$ for a network, and a flow f, to produce a set of k predicted links $\mathcal{L}_p^f = \{l_1, \ldots, l_k\} \subseteq \mathcal{L}$, which maximizes the probability: $P(\text{ingress}(f) \in \mathcal{L}_p^f)$. Specifically, we allow for the selection of multiple links as a knob that operators can turn in order to trade off prediction accuracy and operational utility. Increasing the parameter k can increase prediction accuracy but provides less actionable information. In addition, a flow may enter the network at multiple links (due to load balancing, or multiple flows being combined into a flow aggregate, or a flow shifting over time), and reducing k will limit how many of these multiple links can be predicted, with the probability value predicting what fraction of the flow's bytes will arrive on that link.

3.2 Feature engineering for learning

We can view the traffic ingress prediction problem as a learning problem. Given real observations about where traffic has entered the network in the past, we use this information to build a *model* of WAN ingress to make predictions about where future or unseen traffic will arrive. As noted before, TIPSY does not attempt to predict the future traffic matrix, which is a complementary problem.

A key design issue with any learning task is choosing the right features for the problem. In this work, features must be derived

[2]We use the shortest, valley-free [37] route in the AS-level graph inferred from our BMP data. The source AS may not actually use this path to reach the Azure WAN, but for the purposes of Figure 2, this is a useful approximation.

from the sampled ingress traffic and combined with information about the network. We focus on features related to the IP layer, (e.g., source prefix) as they are the most likely to influence routing decisions.[3] Specifically, we use the following features:

- **Source AS:** AS number where the packet originated, based on the source IP address of the sampled flows and BGP advertisements observed at the collecting router.
- **Source prefix:** All traffic entering the Azure WAN is from a source IP address on the Internet. Using the entire /32 bits of IPv4 addresses significantly increases the feature space. Instead, aggregating to the announced routing blocks sizes may hide useful information, especially if BGP Route Aggregation is enabled by ASes. In this trade-off between resolution and feature space, we use the /24 prefix of the source IP as the feature. Our intuition is that /24 is the widely accepted limit on routable prefix length and inter-domain routing policies could operate at that boundary, which would influence which peering link traffic arrives on.
- **Source location:** The source prefixes feature has a large feature space – we see over 13M /24 prefixes in our dataset. Therefore we choose to also use mappings from source IP addresses to coarse geo-location (at the level of large metropolitan areas). Our intuition is that traffic originating in the same AS and geographic area with the same destination may share paths.
- **Destination region:** All traffic entering the Azure WAN is destined to an endpoint in our network, such as a datacenter server. We use the geographic location of the destination within our network.
- **Destination type**: This is the type of service provided by the destination server (e.g., web service, storage). Our intuition is that load balancing at the application layer (e.g., DNS) may influence how traffic shifts, and this behavior may depend on the destination service.

We always use source AS (A) and both destination region and type, and explore combinations with source prefix (P) and source location (L). Table 1 shows the resulting combinations, together with the number of unique values of the features at the bottom, and the number of unique combinations on the right. Since there is only one location per /24 prefix in our dataset, the APL combination is equivalent to the AP combination, and we do not have to look at it specifically. However, training a model without source prefix but with source location (AL) allows for transfer learning between flows that are arriving from the same geographic location, with a different prefix. In our dataset, there are 564 unique source locations and over 13.5M unique /24 source prefixes. §4.2 explains how we encode these features.

3.3 Model selection

The next design challenge is to build learning models that fit our problem definition and feature space, without unnecessary complexity or long query time when addressing ongoing congestion. As such, there are several relevant design considerations:

[3]ECMP and more sophisticated routing such as that proposed in SDX [19] can use transport and application layer features such as protocol and port, but we leave it as future work to look for evidence of their widespread influence on AS-level paths.

	Source			Destination		
Name	AS	Prefix	Location	Region	Type	#
A	•			•	•	$\sim 9M$
AP	•	•		•	•	$\sim 400M$
AL	•		•	•	•	$\sim 15M$
#	$\sim 72K$	$\sim 13.6M$	~ 600	$\sim 125K$	~ 200	

Table 1: Feature sets used for the models, along with the approximate number of unique items for each feature ('#' line) or combinations ('#' column). Note that because in our network there is only one source location per prefix, there is no need to consider APL, as it is equivalent to AP.

- **Large feature space:** there are over 70K ASes and over 125K internal destination locations in the dataset (defined in §4.1). These features are categorical, with no obvious way to reduce their dimensionality. Furthermore, there are over 13M source /24 prefixes. While the IP address space is hierarchical, it also has growing fragmentation, and potentially arbitrary discontinuities. The large categorical feature space makes it difficult to use certain classification algorithms such as those based on decision trees (e.g., random forest). In our early tests, using these features in a random forest model generated very large and deep trees, which were brittle and impractical.
- **Large number of classes:** there are thousands of peering links in the network, each of which represents a separate prediction class. A statistical classification model must assign, to each such peering link, a probability for it being used as the ingress point for a given flow.
- **Dynamic/evolving network:** the Azure WAN's peering with other networks changes over time, peering between other distant ASes also evolves, and routing policies at these networks also change over time. All of these impact the routing behavior of Internet flows. It is important that the learning models be (re)trained quickly, even taking into account the large feature space and the number of classes.

3.3.1 Models. Given the above design considerations, and after testing several techniques including DNNs (of different depths and widths), we converged on two types of simple statistical classification models that combine fast, single-pass training with small memory footprint, support explainability, and offer accuracy that is close to that of an oracle (§5): Naïve Bayes models and a class of Historical Models. Both also allow for weighting of training samples by traffic volume. Weighting training samples is necessary to (1) provide relative importance to larger flows within a flow aggregate as defined in Table 1, (2) ignore a random stray packet of a flow that arrives on a very different peering link; (3) predict what fraction of a flow will arrive on one link, and what fraction on a second link, and so on; and (4) make adversarial attacks against the models more challenging. Because the results from the historical model dominate those from Naïve Bayes, here we describe and evaluate the historical model, and leave Naïve Bayes for Appendix A

Historical models: Our intuition is that Internet routing policies are (usually) slow-changing and hence the likelihood that traffic with the same characteristics will arrive at an ingress link that was used in the past is high. With this in mind we design a Historical

model (Hist) that relies on arrivals in training data to predict the ingress link for a flow.

Let $B(\cdot)$ represent the total bytes seen in training for a given condition $(\cdot)$. For example, $B(AS = 3, l = 129)$ is the number of bytes seen on all flows from AS 3 on link 129. For a given flow $f = \langle f_1, f_2, \ldots, f_n \rangle$, represented by a subset of the features above (e.g., f_1 = source AS, f_2 = source prefix, etc), the model computes the probability that ingress(f) = l, which we write as $p(l|f)$:

$$p(l|f) = \frac{B(f,l)}{B(f)}$$

Given this definition, $\mathcal{L}_P^f$ is then simply the set of the top-k peering links sorted by $p(l|f)$.

We train a Historical model by (i) collecting all the unique combinations of features seen during training (feature tuples), (ii) finding all the ingress links used for each tuple, and (iii) ranking the peering links according to how much traffic belonging to the tuple was measured. We use the ranks of the links as priorities when using the model to predict a peering link.

The most significant limitation of the historical model is that there is no transfer learning between tuples: if an ingress link was not traversed for a given tuple during training, it cannot be predicted for that tuple. This limitation also means that if a tuple was not seen during training the model cannot predict an ingress link. However, on the flip side, this limitation allows the model to prevent independent flows from affecting each other. However, the larger the flow aggregate, such as in AL, the more specific flows will comprise it, and we expect some transfer learning between those specific flows within a single flow aggregate.

We trained and evaluated three Historical models, according to the feature sets from Table 1: Hist_A, Hist_{AP}, Hist_{AL}. Because there is only one location per prefix, Hist_{APL} is equivalent to Hist_{AP}.

Ensemble models: We also trained and evaluated sequential ensembles of the above models, to leverage their complementary strengths. In our notation, A/B means sequential composition of the models: we resort to model B if there is no prediction for a flow in model A. We examined the following ensemble models: $\text{Hist}_{AP/AL/A}$, and $\text{Hist}_{AL/AP/A}$. In doing so, our motivation is to overcome the limitation of transfer learning in historical models. A model that uses a more specific flow definition will first provide predictions, and subsequently a model with a less specific flow definition will provide predictions. In this way, our hope is the most accurate prediction is provided first, followed by those that incorporate more transfer learning. For this reason we use ensembles sequentially instead of other schemes such as majority voting.

Geographic distance of peering: Finally, there are flow aggregates for which we do not see k alternative ingress links, even though they may exist. In these cases, we added a simple strategy that uses geographic distance to find alternate peering links. We take the peering AS A and ingress location l for the best match ($k = 1$), and rank the other peering interfaces from A by geographic distance to l. The model then uses this ranked list to complete the list of interfaces returned. We evaluated this strategy on top of the AL models (as it was the best for unseen withdrawals), and call it $AL + G$ in the rest of the paper.

Table 2 summarizes the complete set of models we used.

Type	Name	Features
Historical	Hist_A	AS, Dest
	Hist_{AP}	AS, Prefix, Dest
	Hist_{AL}	AS, Source Location, Dest
Ensemble	$\text{Hist}_{AP/AL/A}$	Hist_{AP} then Hist_{AL} then Hist_A
	$\text{Hist}_{AL/AP/A}$	Hist_{AL} then Hist_{AP} then Hist_A
+ Distance	Hist_{AL+G}	Hist_{AL} + Geographic Distance

Table 2: Summary of the models trained and evaluated in this work. Dest means both destination features.

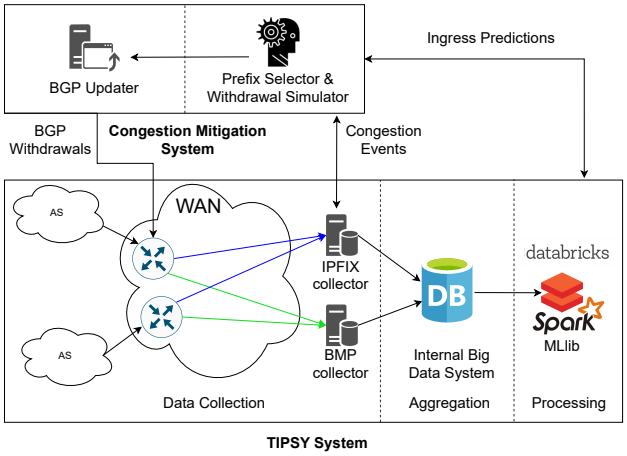

Figure 4: Overview of TIPSY architecture

4 SYSTEM OVERVIEW

We have built and integrated the learning models from §3.3 into a system called TIPSY. Figure 4 shows the high-level architecture, consisting of several components including data collection, aggregation / pre-processing, learning / inference, and integration with a congestion mitigation system (CMS).

We designed TIPSY to run online as a prediction service and to retrain its models daily. Given the timescale of the actions that TIPSY enables, we found this to be a good trade off of accuracy and cost – this allows it to make fast predictions for the congestion mitigation system while also maintaining accurate prediction models and reducing compute cost.

When asking TIPSY for a prediction, it is given a set of traffic flows (tuples and bytes) arriving at the prefixes that will be withdrawn, and at which peering links they will be withdrawn. TIPSY operates on those input flows as is and does not predict future traffic volumes. In practice, the list of flows and volumes from recent past peak times tends to be sufficient.

4.1 Data collection

We utilize three pieces of data.

(1) BMP: We use the BGP Monitoring Protocol (BMP) [4] data collected from the Azure WAN. BMP exports all the announcements and withdrawals that a router has received from any neighbor, even if those advertisements do not impact the forwarding table.

BMP data listeners are spread across the Azure WAN, connect to BGP routers including all peering routers, store advertisements in a large data lake, and use automatic mechanisms to recover from failures. TIPSY does not use this data to train any models nor execute any models. Rather, this data is used for debugging purposes and producing non-operational analysis such as Figures 2 and 3. We do not use BMP data from other ASes, as those are not available to us.

(2) IPFIX: We use IP Flow Information Export (IPFIX) [3] data collected from the Azure WAN. IPFIX exports flow level information for data-plane packets that traverse a router interface. Due to the massive amount of traffic traversing the Azure WAN, IPFIX data collection is configured to sample 1 out of every 4096 packets at random. While that reduces the likelihood of capturing short-lived or low throughput flows, all of the use cases for TIPSY concern shifts in large volumes of traffic. Similar to BMP, the Azure WAN has distributed collectors that consolidate the flow data into a conceptually central data lake. TIPSY processes only the ingress flows recorded at all the peering routers. The important fields in the IPFIX data that we use include source address prefix, source ASN, destination address prefix, timestamp, and number of bytes scaled up by the sampling rate.

(3) Metadata: We also rely on network metadata to augment our BMP and IPFIX data. The destinations of all the IPFIX flows that TIPSY processes are all inside the Azure WAN. We identify which cloud service (e.g. storage or e-mail) and which metropolitan region the flows are destined to. We use a comprehensive internal Geo-IP database to identify in which country and metropolitan region the external source prefix originates. BMP and IPFIX data have the identity of the router and interface that the data was collected from, and we look up which peer that link connects to and in which geographic region it is.

Explicit attempts at ingress traffic engineering by altering outbound BGP route announcements (e.g., by AS path prepending) can alter the "normal" flow of ingress traffic. Such human-induced meddling could have adverse effects on the prediction accuracy of TIPSY. For the time periods we evaluate in this paper, we identified all the prefixes that were involved in any kind of ingress traffic engineering. These prefixes accounted for 0.7% of all the prefixes announced by the WAN and none of these prefixes were present in the destination addresses in the IPFIX flows. We believe that these prefixes were used for measurement experiments with minuscule amounts of traffic. As part of future work, we plan to consider explicit signaling from ingress traffic engineering systems to TIPSY to enable better predictions for such traffic, if the volume of such traffic becomes significant.

We have trained and evaluated TIPSY on many time windows across many months and observed consistent accuracy. In this paper, we present results from 10 November 2021 to 6 December 2021 for training and testing our models.

4.2 Data aggregation

We store all our data in a large, geographically-distributed data lake that is accessible to our compute systems. The volume of data is large – the BMP and IPFIX data each consume TBs of storage per day. To allow our learning system to train and test on weeks of data, we have implemented a multi-stage pipeline. We aggregate and compress the data prior to feeding it to our learning system. We aggregate the IPFIX data into hour long chunks, indexed by only the features that TIPSY uses (defined in §3.2).

Aggregating flows into hour-long chunks does not limit the ability of TIPSY to predict for flows at smaller or larger time windows, as the aggregation merely sums the bytes received for a flow. Aggregating flows by some features does limit TIPSY to not being able to distinguish between two flows that are identical in those features but different in features that we ignore (such as port numbers). However, for the purposes of addressing ingress peering link congestion, that fidelity serves no purpose in calculating how many bytes will shift to where, as a result of prefix withdrawals.

We join this IPFIX data with metadata to add additional salient information as previously described. We compress the features in this data by using a simple dictionary (i.e., ordinal encoding).

We run the aggregation for TIPSY on a performant, internal, big data processing system that is optimized to scale for relatively simple tasks, such as filtering and joining, but is not flexible enough to support more complex tasks such as machine learning. We implemented the aggregation component of the pipeline in an SQL-like framework. After aggregation, the IPFIX data is 2% of the original size and the BMP data is 0.07% of the original size.

Model	Training	One prediction	Model size
Hist_A	$O(n)$	$O(1)$	$O(\|A\|)$
Hist_{AP}	$O(n)$	$O(1)$	$O(\|AP\|)$
Hist_{AL}	$O(n)$	$O(1)$	$O(\|AL\|)$

Table 3: Model costs in terms of training, prediction, and memory complexity. n is the number of data points; l the number of peering links; $|A|$,$|AP|$, and $|AL|$ are the number of unique combinations of the features defined in Table 1.

4.3 Model execution

To train and execute the learning algorithms from §3.3, we use a dedicated Databricks cluster, consisting of 6 worker nodes with 640GB RAM and 80 cores each and NVMe storage, and a driver node with 256GB RAM and 64 cores. We use the Databricks 10.2 runtime, which includes Apache Spark 3.2.0 and Scala 2.12.

The models we selected only require one pass on the data to train. Table 3 summarizes the runtime and memory costs of training and prediction with the different models we use.

When training the historical model, it is necessary to group all the measured ingress traffic by the respective flow tuples (as defined in Table 1) and ingress link, which requires memory and processing linear with the number of measurements (after the aggregation stage). Then for each flow tuple the peering links are ranked by bytes, keeping only the top k links. This requires memory and processing linear with the number of unique flow tuples in the data (the number of peering links per tuple is relatively very small). The resulting historical models are linear in size with the number of unique flow tuples in the training data. Using the historical model for predictions is essentially a lookup in the model table, which in Spark is done efficiently as a join between the list of flows of interest and the model table.

The memory cost of the ensemble models is the sum of the costs of the components models; their runtime is a weighted sum of the runtimes of the component models, where the weights depend on whether the queries are satisfied by an earlier model.

4.4 Congestion Mitigation System

TIPSY is integrated with an in-house congestion mitigation system (CMS) that leverages traffic ingress predictions to strategically shift traffic away from congested peering links.

CMS monitors ingress peering link utilization. When it observes that traffic volume has exceeded a fixed threshold (greater than 85% link utilization for at least 4 minutes), it identifies in the IPFIX data the fewest destination prefixes (top prefixes by traffic volume) at the affected peering links that if shifted, would bring utilization back down to acceptable levels.

Several tradeoffs are involved in picking the triggering utilization threshold and time window. They include wanting to detect congestion quickly, avoid unnecessary withdrawals if the congestion is very short lived, and limit how much traffic is dropped or delayed. In practice, 85% utilization for at least 4 minutes has worked well, but both parameters can be easily modified.

CMS considers prefixes at the same size granularity as what is being advertised in our BGP sessions. In theory, it could break apart a large prefix into smaller prefixes and selectively withdraw some of them, but does not do so in an effort to balance the tradeoffs between fine-grained TE, increasing the BGP table sizes of our neighbors, and route convergence time.

CMS then queries TIPSY to determine which other peering links these flows would likely shift to if CMS were to withdraw each of those prefixes. With these predictions, CMS determines which prefixes are safe to shift, will not overload other peering links, and yet will shift sufficient traffic away from the congested peering link. CMS then injects BGP withdrawal messages into the edge router for these prefixes. Later, when traffic volumes have returned to normal, those prefixes are re-announced at the original peering link.

Prior to TIPSY, CMS would issue a withdrawal for prefixes, not knowing which other links would receive their traffic, and if those links had capacity. When new links became overloaded, CMS would issue additional withdrawals on those other links, just as in the example in §2.

5 EVALUATION OF TIPSY

In evaluating TIPSY, our goal is to address several questions that are pertinent to a WAN operator:

- How do we define the prediction accuracy of these models?
- How accurate and close to optimal are different models?
- How accurate are different models at predicting traffic ingress during normal operations vs. for BGP withdrawals?
- How long should the training window be?
- How long is a trained model useful for?
- What is the best model, considering accuracy, generality, and complexity?
- How effective is TIPSY in identifying at-risk peering links?

5.1 Methodology

5.1.1 Training data. Ideally we would evaluate the accuracy of TIPSY on withdrawal events, and verify if the bytes from affected flows would indeed enter the network through the interfaces predicted by TIPSY. If we were to evaluate the accuracy of TIPSY's predictions on the outcome of only those prefix withdrawals that CMS determines to be safe based on TIPSY's predictions, it could bias the results. It would eliminate from the accuracy calculation those withdrawals that were deemed unsafe and not executed by CMS on the basis of predictions by TIPSY.

Instead, we examine another source of prefix withdrawals that occur in network operation – peering link outages. Peering link outages are similar to local prefix withdrawals since the BGP session is terminated and traffic must seek alternative routes into the Azure WAN. However, unlike with individual prefix withdrawals, peering link outages provide a massive source of additional data as to where ingress traffic will be redirected since all advertised prefixes are effectively withdrawn for the duration of the outage. We observe that over a year most links have at least one outage (§5.3).

To evaluate TIPSY, we examine previous peering link outages that last for any contiguous period between 1 hour and 24 hours. We do not consider outages that last longer than 24 hours as those tend to represent exceptional cases (e.g., decommissioning peering, or natural disaster). We do not consider outages that last less than 1 hour because TIPSY processes massive quantities of data and one of the techniques we use to scale is to aggregate data into hour-long windows before it reaches the training component.

We infer outages from IPFIX data – if a peering link received no bytes in a one-hour window, we consider it to have an outage. While using IPFIX data to find outages may not seem intuitive, it is the ground truth about the operating state of the network. We found that other sources, such as SNMP, were far less reliable. Both for training and evaluation, for each flow (or aggregate) that was in a link with an outage, we look at all the links where the flow appears during the outage.

For our evaluation we use three-week training periods of ingress traffic data for TIPSY to learn from and test on one week of unseen traffic ingress data. For our main results we used training data from November 10–30, 2021, and tested on traffic data from December 1–6, 2021. The justification for three-week training and one-week testing time periods were derived empirically (see Appendix B).

5.1.2 Prediction accuracy. As explained in Figure 3, we have observed traffic from some source ASes entering the WAN at as many as a few hundred different peering links. Given some aggregate of traffic from Table 1 in the testing data, and the need to predict where that traffic will arrive on the WAN in the future, the individual predictions are numerous. A model would identify which peering link will receive the most bytes, followed by which other link will receive the second most bytes, and so on for hundreds of links. The accuracy of a model is the sum of all the bytes that it correctly matched to the actual links that received the traffic in testing data, divided by the sum of all the bytes for all flows.

However, training a model to provide accuracy at the hundredth and beyond possible link that can receive traffic for a flow tuple, and calculating such predictions is both computationally inefficient and unnecessary. While traffic from an AS may be spread across many

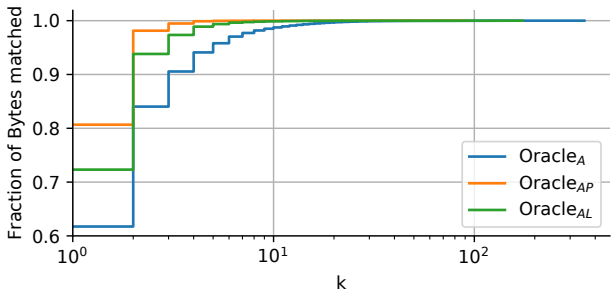

Figure 5: Prediction accuracy of oracle as a function of the number of ingress links predicted.

peering links, individual flows might arrive at fewer links. The CMS needs to know which peering link(s) will receive large volumes of traffic in different situations, and small amounts of bytes are immaterial to that need. How many links per ingress flow should our models be trained for and evaluated against?

We answer that question with Figure 5. We define an oracle as one that has perfect knowledge of the testing data - it knows exactly which link receives how many bytes for every ingress flow. However, we limit it to providing a maximum number of predictions per flow. For example, when k=10 in Figure 5, the oracle gives the identity of the link and the number of bytes it received, for each of the ten links that receive the most traffic for each ingress flow in the testing data. We calculate the accuracy of the oracle for each of the three definitions of tuples from Table 1.

At the tail end of graph, where the oracle is unrestricted, it provides perfect prediction accuracy of 100%. When restricted to predicting how many bytes arrive at only the top 1 link for a flow (the left most point), it can provide only 65-85% accuracy and misses what happens to 15-35% of the traffic. Worse, if that one link for a flow is unavailable (due to a link outage or prefix withdrawal), it can provide no answer.

We pick the top-3 link prediction accuracy as the main metric. At k=3, the Oracle$_{AP}$ and Oracle$_{AL}$ versions of the oracle demonstrate that 97% of bytes are theoretically predictable. Later, we present prediction accuracy results for k=1, k=2, and k=3, with the target benchmark being k=3. Note that this top-3 prediction metric does not mean that a model gets three guesses and if one of them is correct, then it gets perfect accuracy. Rather, to get perfect accuracy, it has to predict exactly how many bytes arrive at each of the 3 links that will receive the most bytes for a flow, and there should be no other bytes arriving on any of the other links. Also note that Figure 5 shows the CDF across every individual flow – the total number of peering links that a model needs to consider is still large.

5.2 TIPSY prediction accuracy

Table 4 shows the prediction accuracy for future traffic for the different models and the oracles. Our target benchmark is Top 3 ($k = 3$) accuracy as a percentage, but we include the two lower values for k as well. The main conclusion is that accuracies for the *AP* and *AL* models are very high - above 93% in all cases, and close to the relevant oracle accuracy. The accuracy results for January 2021 in the Appendix (§D) are even higher (closer to 98%).

Model	Top 1 %	Top 2 %	Top 3 %
Oracle$_A$	61.74	84.03	90.55
Hist$_A$	59.36	82.07	89.02
Oracle$_{AP}$	80.66	98.13	99.46
Hist$_{AP}$	75.62	95.28	97.09
Oracle$_{AL}$	72.31	93.81	97.34
Hist$_{AL}$	69.62	91.85	95.73
Hist$_{AL+G}$	69.62	91.93	95.86
Hist$_{AP/AL/A}$	**76.02**	**95.95**	**97.88**
Hist$_{AL/AP/A}$	69.64	91.87	95.76

Table 4: Overall prediction accuracy, with 3 weeks of training and 1 week of testing. Numbers in bold show the best accuracy for each k across all models.

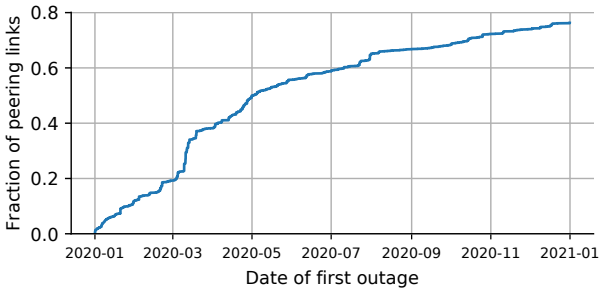

Figure 6: Earliest time in calendar year 2020 that a peering link was down.

5.3 Prediction accuracy for withdrawals

Given our use of peering link outages as a proxy for BGP route withdrawals from §4, we collected data to better understand the frequency with which peering link outages occur for the Azure WAN. Specifically, we used IPFIX data collected over the entire calendar year for 2020 and inferred peering link outage events based on our description in §4. From this data, we made two observations:

(1) Most links have an outage at least once in the year: As seen in Figure 6, the rate of new outages (outages on peering links that did not experience a outage previously) grows almost linearly over time and covers about 80% of all the peering links that were active during that period. Note that we include all peering links, both direct peering as well as Internet exchanges. The vast majority of our peering links are direct peering.

(2) Some links had a recent outage: To complement the analysis of the first time an outage was seen we also checked the last time an outage was seen (Figure 7): looking back from the first day after the period, how long ago was the last time the link was down. Like in the forward direction, looking back we see a mostly even spread of unique outages over time with roughly a third of links experiencing an outage within the previous 50 days.

In summary, peering link outages are an operational fact - they happen frequently. Despite their frequency, we cannot assume that past history is a perfect template of what will happen in the future, for two reasons. First, we would have to go far back in history to capture all link outages, even further back than the data we

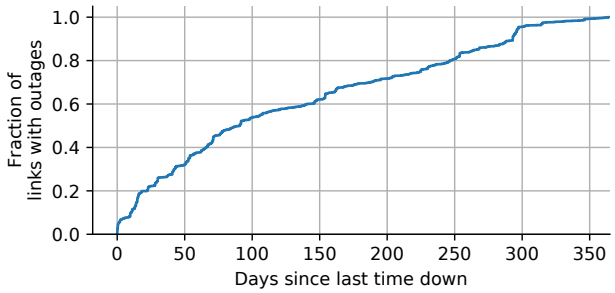

Figure 7: Days since the last time a peering link was down.

Model	Top 1 %	Top 2 %	Top 3 %
Oracle_A	78.67	86.16	92.35
Hist_A	55.69	62.92	67.45
Oracle_{AP}	94.25	98.41	99.56
Hist_{AP}	58.93	62.88	64.08
Oracle_{AL}	86.04	93.4	97.33
Hist_{AL}	60.74	67.54	70.65
Hist_{AL+G}	62.71	**71.12**	**76.42**
$\text{Hist}_{AP/AL/A}$	**64.64**	70.18	73.44
$\text{Hist}_{AL/AP/A}$	60.84	67.73	71.58

Table 5: Prediction accuracy, all link outages: with 3 weeks of training and 1 week of testing, only for traffic that would have arrived on a failed peering link. Best results in bold.

currently have which is for 1 year. IPFIX data for our network consumes PBs per day, and is growing rapidly. To store a year's data on high performance storage that is replicated across compute clusters for such analysis, would cost almost $10M USD annually at public cloud rates, and even that is insufficient to capture all outages. Second, even if we capture all the necessary data, if a link outage happened at a low load time for the link, it may not reflect what would happen at a peak time. Furthermore, traffic patterns for the Azure WAN change over time.

5.3.1 All withdrawals. We now examine the accuracy of TIPSY models specifically for traffic in the testing periods that were impacted by BGP withdrawals initiated by peering link outages. Here, we consider only those flows where the top 1 link that received traffic during training was unavailable during testing, and only that duration of time during testing when the link was unavailable. To evaluate each model, it is given the prior of which peering links are unavailable for which time periods. Hence those are not valid choices, and the model will provide which link has the next highest likelihood of receiving traffic. The prediction accuracy is shown in Table 5. From the table, we make the following observations for top $k = 3$ prediction accuracy:

- Models using source location (AL) outperform those without (A, AP) – i.e., Hist_{AL} is more accurate than Hist_A and Hist_{AP}. The AL models allow for better generalization than AP, and make effective use of more information than A models.

Model	Top 1 %	Top 2 %	Top 3 %
Oracle_A	82.04	89.34	92.69
Hist_A	77.25	82.82	85.42
Oracle_{AP}	95.59	99.01	99.89
Hist_{AP}	88.02	91.08	92.52
Oracle_{AL}	90.15	96.35	98.52
Hist_{AL}	84.49	89.61	91.97
Hist_{AL+G}	84.62	89.77	92.43
$\text{Hist}_{AP/AL/A}$	**89.25**	**92.82**	**94.57**
$\text{Hist}_{AL/AP/A}$	84.52	89.66	92.04

Table 6: Prediction accuracy, seen outages: 3 weeks of training and 1 week of testing, only for traffic that would have arrived on a peering link with an outage, where the outage was also experienced during training. Best results in bold.

Model	Top 1 %	Top 2 %	Top 3 %
Oracle_A	76.14	83.78	92.09
Hist_A	39.52	47.99	53.97
Oracle_{AP}	93.25	97.97	99.31
Hist_{AP}	37.1	41.73	42.75
Oracle_{AL}	82.95	91.19	96.44
Hist_{AL}	42.92	50.99	54.66
Hist_{AL+G}	**46.33**	**57.31**	**64.56**
$\text{Hist}_{AP/AL/A}$	46.17	53.2	57.6
$\text{Hist}_{AL/AP/A}$	43.07	51.27	56.23

Table 7: Prediction accuracy, unseen outages: 3 weeks of training and 1 week of testing, only for traffic that would have arrived on a peering link with an outage, where the outage was not experienced during training. Best results in bold.

- Using geographical heuristics to complement learned traffic patterns improves the accuracy significantly. Hist_{AL+G} performs better than the other models – hot potato routing is not uncommon for outages.
- The best models are Hist_{AL}, Hist_{AL+G}, $\text{Hist}_{AL/AP/A}$, and $\text{Hist}_{AP/AL/A}$.

Even though geolocation can be imprecise [31], it is quite useful in TIPSY. For the purposes of learning the behavior of hot potato routing policies, precise geolocation is often not necessary – it need only be as precise as the difference in IGP weights between alternate paths on an AS'es backbone, and in our experience, metro-level precision is sufficient. We rely on a proprietary Microsoft geolocation database that aggregates information from many sources [12]. It has sufficiently broad coverage and precision for TIPSY. Finally, in Hist_{AL+G} which relies on the location of peering links, we know precisely where every peering link on the Azure WAN is, down to the exact street address.

5.3.2 Seen and unseen withdrawals. To dissect the accuracy of our models for traffic impacted by BGP withdrawals due to link outage periods, we split the results into seen and unseen link outages. Table 6 shows the accuracy for only those flows and time windows where the primary receiving link had an outage in testing, and the same outage was also experienced during training. In contrast, Table 7 shows the accuracy for only those flows and time windows

where the primary receiving link had an outage in testing, and the same outage was not experienced during training. The number of outages experienced during this time window are in line with Figures 6 and 7. The total bytes affected by unseen outages was about 57% of total bytes affected by all outages. We find that:

- TIPSY can predict the ingress for traffic experiencing a withdrawal that was not previously experienced with surprisingly good accuracy – about 65% for the best model. Despite the withdrawal not having been experienced during training, even under normal circumstances many flows arrive on multiple peering links over time, either due to load balancing or other routing variations on the Internet. Hence there is opportunity for models to learn other likely links that flows will arrive on.
- Source location was not helpful for predicting traffic ingress during seen link outages, however was quite useful in unseen outages. We believe this is because for seen outages, past behavior of how flows were re-routed is still valid, especially since we are using only a 3-week training window, and hence more specific AP models do well. For withdrawals caused by unseen link outages, the geographic location of the source is useful in AL models as it accounts for hot potato routing in picking the next best link.
- This is also why the most accurate model is different between Table 4 and Table 5. In the former case, most traffic continues to arrive on the same links as previously observed, while the majority of traffic in the latter case involves unseen link outages.
- Geographic heuristics are effective for unseen outages. This provides strong evidence for hot potato routing in peering networks.
- Considering the accuracy for traffic impacted by both seen and unseen link outages and overall accuracy for top $k = 3$, simplicity of implementation, and training and execution overhead, we consider the best TIPSY models to be Hist_{AL+G} and $\text{Hist}_{AP/AL/A}$. We expect that for higher values of k, longer ensembles may provide more value.
- These accuracy results are similar to what we see in other time periods, some of which can be found in the appendix.

6 EXPERIENCE WITH TIPSY IN AZURE

As explained in § 5.1.1, we focused our quantitative evaluation on peering link outages rather than prescribed BGP withdrawals. Nonetheless, we have examples of TIPSY's accuracy in real prefix withdrawal incidents. §2 described one. Here we describe a few more. In each case, we use a 3-week training window prior to the incident with Hist_{AL+G}. The accuracy of TIPSY in peering link outages and these incidents below show that it is effective for both wholesale route withdrawals as well as selective route withdrawals.

On 06 September 2021, a peering link in East Asia hit high utilization. CMS withdrew two /24 prefixes. As a result, thousands of flows were shifted, with about a hundred that were over 10Mbps each. The goal was to shift about 10Gbps – just enough traffic to bring down utilization to acceptable levels on this peering link, without shifting too much traffic. In post-incident analysis, we ran a TIPSY model trained on traffic prior to the incident. It identified three links that the traffic would shift to, with two different transit providers, two in the same metropolitan region and one in a different country in East Asia, with the first two links absorbing the majority of the traffic. All three links had sufficient capacity to absorb the traffic. After CMS issued prefix withdrawals, traffic shifted as predicted to those links. 2 hours after the withdrawals, traffic levels had dropped sufficiently that the prefixes were re-announced by CMS.

On 15 November 2021, a similar incident occurred in a different peering link in East Asia. Again, in post-incident analysis, flows impacted by prefix withdrawals were correctly identified by TIPSY as shifting to other peering links, both in the same metropolitan region and another country, involving multiple ASes including transit providers.

Earlier in 2021, in northern Europe, a peering link was congested in the ingress direction. CMS issued prefix withdrawals, but this caused another peering link with the same peer in the same location to become overloaded. CMS then issued additional withdrawals from the second link resulting in other peering links with the same peer in a different city to become overloaded. After a third round of prefix withdrawals, traffic was sufficiently spread out to other peering links with the same peer in other countries. This series of congestion incidents lasted about 2 days. Even though TIPSY was not deployed at the time of this incident, we later ran it on the affected flows and interfaces. TIPSY correctly identified the cascade of traffic shifts on multiple links across the three withdrawal incidents. If CMS were armed with this knowledge, it could have issued prefix withdrawals simultaneously on all three sets of links from the very start, reducing the congestion window.

In our experience, we have not noticed any side-effects of CMS's prefix withdrawals, such as other ASes doing egress TE in reaction to our BGP withdrawals. Once traffic has calmed down, CMS re-announces those prefixes. Since these congestion events last for minutes to a few days, the withdrawals do not cause long term impact to capacity planning.

As shown in Table 5, there are situations where TIPSY does not correctly predict where a flow will arrive. In our experience running TIPSY, we have observed this happening in two types of situations:

- The flow has previously arrived on one of several links, with no discernable pattern to identify which of the several links it will arrive on. The actual link it arrives on during an outage is in a $k > 3$ prediction position.
- The flow has previously arrived on only one link, other flows similar to it (same source AS and location and destination) have also arrived on only that link, and that link is the only peering connection with the neighbor AS.

The first situation requires additional work, such as considering a larger k, which comes at the risk of making the decision by CMS more complicated. The second situation should resolve itself over time, as TIPSY observes the flow's behavior when a prefix withdrawal is issued.

There are two additional scenarios that are possible, which we have not yet observed:

- There are significant changes to the topology or routing policy of a neighbor AS.
- A neighbor AS has only one peering link with the Azure WAN and uses default routing, in that it will always send all traffic to that specific peering link.

In the first scenario, we again expect that TIPSY will recover in time as it observes new routing behavior. When running in the Azure WAN, TIPSY is re-trained daily to incorporate the latest observed flow arrivals. In the second scenario, TIPSY will have no prediction

if CMS withdraws prefixes on the peering link, which is the correct answer as there is no alternative path. Note that if some distant AS uses default routing, but intermediate ASes do not, then TIPSY will still provide useful predictions as those intermediate ASes still have multiple paths to the Azure WAN that they can use.

Given how CMS uses TIPSY (§4.4), the penalty of an incorrect prediction is low. If TIPSY predicts too few flows on a link, CMS may have to issue additional withdrawals later, no different than what it did prior to TIPSY. If TIPSY predicts too many flows on a link, it will eliminate what should have been a safe withdrawal by CMS, leading it to choose a different safe withdrawal. In the worst case where all withdrawals are deemed unsafe, CMS has no choice but to revert back to its original behavior prior to TIPSY.

7 PRIOR WORK

To the best of our knowledge, TIPSY is the first work to both define the problem of traffic ingress prediction, leverage its use in congestion mitigation and capacity planning, and to solve it using a learning approach. Nonetheless, there are several lines of work related to TIPSY, namely topology modeling, traffic engineering (TE) and traffic matrix (TM) prediction, which are complementary to the goal of predicting ingress traffic locations.

Topology modeling: Many previous works have addressed the problem of predicting Internet paths [14, 26, 28, 29, 36], usually through a combination of traceroute measurements and BGP topology information. TIPSY is concerned with the related but different problem of predicting the ingress interface to a single large WAN, not the entire paths. It also does not have access to AS paths from the point of view of the *sources*. More importantly, TIPSY uses all observed traffic as training data to predict likely volumes of traffic in different possible interfaces, overcoming biases from BGP data, and insufficient coverage from data plane measurements [10].

Traffic engineering: Pujol et al. [32] propose mechanisms for eye-ball networks to inform peer content-provider networks about the best ingress links to reach internal customers, based on their internal topology. This is complementary to the problem of *predicting* how arbitrary sources will send traffic in current and potential network conditions. Such mechanisms are also not sufficient, as they would not cover ASes beyond the first hop. Further, given the number and diversity of peer ASes to the Azure WAN, it likely that many such peers would not participate.

Researchers have extensively studied the problem of TE in the context of WANs. Solutions are divided into *offline* and *online* systems depending on the frequency of re-optimization. Offline ones include TEAVAR [9], INITE [17], COPE [39], and METL [6]. Online systems can be further classified into centralized vs. distributed solutions. TE solutions such as the proposal by Uhlig and Bonaventure [38], RCP [11], B4 [21], SWAN [20], Espresso [40], and Edge Fabric [35] optimize the network from a centralized vantage point, while other solutions, e.g., MPLS-TE [8], TexCP [23], MATE [16], and HALO [27], perform distributed TE.

While TE solutions may improve network performance (latency, utilization), these systems rely on measured or predicted traffic information (*e.g.*, traffic matrix, application demand, *etc.*). TIPSY is complementary to TE work in that it predicts wide-area network ingress traffic locations, which is useful for TE, among other tasks.

Researchers have also proposed TE with migration [24] using a technique called "router grafting". It would allow a network to migrate BGP peering sessions to other locations and thereby alter the TM to a more desirable state. Since the remote network is unaffected, and may not even be aware of such a change, there is limited impact to routing upstream. In contrast, when considering BGP withdrawal, routing changes propagate outward leading to the prediction problem that we solve in this paper with TIPSY.

Traffic prediction: Predicting the volume of traffic on individual links or in a TM is yet another complementary line of work [13, 25, 34]. Such techniques are useful for planning to stay ahead of traffic growth. While some use cases may overlap with those of TIPSY, these techniques do not answer "what-if" questions, such as what happens after a BGP withdrawal. Rather than predict demand, TIPSY focuses on the highly challenging problem of predicting per-prefix ingress peering links for large WANs.

8 CONCLUSIONS

Predicting where traffic will enter a WAN is a challenging problem. The size and complexity of the Internet topology, routing policies and preferences employed by many networks, and the large peering surface areas of content provider and public cloud WANs all contribute to making this problem complex. In this paper, we have defined the WAN ingress prediction problem, identified a simple learning-based approach to solve it, and demonstrated 76% accuracy in predicting the 3 peering links that will receive the most bytes of a flow for the Azure WAN after BGP withdrawals. Such accuracy is sufficient to drive "what-if" analysis of candidate BGP route withdrawals and enables TIPSY's use in a congestion mitigation system that safely steers traffic away from congested peering links.

While we focus on this one operational need, there are other useful tasks that can benefit from TIPSY. We can analyze the risk of any single peering link outage or single router or single site outages, described in Appendix C. We have started to use TIPSY to identify suspicious ingress traffic, where it is exceedingly unlikely that a flow would arrive on a peering link. For example, we have identified traffic supposedly from US national labs on peering links in countries far away from the US. Operators could send such spoofed traffic through DoS scrubbers. We could also use TIPSY for de-peering. In the course of maintaining a large WAN, it is natural to consider de-peering to reduce cost and operational overhead with peers that add low value.

Going forward, active WAN manipulation, such as by commercial SDWAN products [1, 2] and by systems such as Google Espresso [40] and Facebook Edge Fabric [35], could increase the diversity of ingress that a flow takes to reach a network. This is an interesting problem that we plan to study in the future. To the extent that such systems are driven by performance improvements, perhaps TIPSY can also incorporate performance data to detect such manipulation and consider performance in identifying the likely ingress.

ACKNOWLEDGEMENTS

We thank our shepherd, Oliver Hohlfeld, and the anonymous reviewers for helping us improve our paper.

REFERENCES

[1] Arista Software Defined Cloud Networking. https://www.arista.com/en/solutions/software-defined-networking.

[2] Barracuda CloudGen WAN. https://www.barracuda.com/products/cloudgenwan.

[3] IETF RFC 7011: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. https://tools.ietf.org/html/rfc7011.

[4] IETF RFC 7854: BGP Monitoring Protocol. https://tools.ietf.org/html/rfc7854.

[5] Microsoft Azure Peering Policy. https://docs.microsoft.com/en-us/azure/internet-peering/policy.

[6] Sharad Agarwal, Antonio Nucci, and Supratik Bhattacharyya. Measuring the Shared Fate of IGP Engineering and Interdomain Traffic. In *Proceedings of the 13TH IEEE International Conference on Network Protocols*, ICNP '05, pages 236–245, Washington, DC, USA, 2005. IEEE Computer Society.

[7] Todd Arnold, Jia He, Weifan Jiang, Matt Calder, Italo Cunha, Vasileios Giotsas, and Ethan Katz-Bassett. Cloud Provider Connectivity in the Flat Internet. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, pages 230–246, New York, NY, USA, 2020. Association for Computing Machinery.

[8] D. Awduche, J. Malcolm, J. Agogbua, M. ODell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702, RFC Editor, 9 1999.

[9] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjørner, Asaf Valadarsky, and Michael Schapira. TEAVAR: Striking the Right Utilization-availability Balance in WAN Traffic Engineering. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, pages 29–43, New York, NY, USA, 2019. ACM.

[10] Randy Bush, Olaf Maennel, Matthew Roughan, and Steve Uhlig. Internet optometry: Assessing the broken glasses in internet reachability. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, page 242–253, New York, NY, USA, 2009. Association for Computing Machinery.

[11] Matthew Caesar, Donald Caldwell, Nick Feamster, Jennifer Rexford, Aman Shaikh, and Jacobus van der Merwe. Design and implementation of a routing control platform. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 15–28, Berkeley, CA, USA, 2005. USENIX Association.

[12] Matt Calder, Manuel Schroder, Ryan Gao, Ryan Stewart, Jitu Padhye, Ratul Mahajan, Ganesh Ananthanarayanan, and Ethan Katz-Bassett. Odin: Microsoft's scalable fault-tolerant cdn measurement system. In *USENIX NSDI*, April 2018.

[13] G. Choudhury, D. Lynch, G. Thakur, and S. Tse. Two use cases of machine learning for SDN-enabled IP/optical networks: traffic matrix prediction and optical path performance prediction [Invited]. *IEEE/OSA Journal of Optical Communications and Networking*, 10(10):D52–D62, Oct 2018.

[14] Ítalo Cunha, Pietro Marchetta, Matt Calder, Yi-Ching Chiu, Brandon Schlinker, Bruno V. A. Machado, Antonio Pescapè, Vasileios Giotsas, Harsha V. Madhyastha, and Ethan Katz-Bassett. Sibyl: A practical internet route oracle. In *NSDI*, March 2016.

[15] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[16] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 3, pages 1300–1309 vol.3, April 2001.

[17] Ruomei Gao, Constantinos Dovrolis, and Ellen W. Zegura. Interdomain Ingress Traffic Engineering Through Optimized AS-Path Prepending. In *Proceedings of the 4th IFIP-TC6 International Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems*, NETWORKING'05, pages 647–658, Berlin, Heidelberg, 2005. Springer-Verlag.

[18] Timothy Griffin and Gordon T. Wilfong. Analysis of the MED Oscillation Problem in BGP. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, ICNP '02, pages 90–99, USA, 2002. IEEE Computer Society.

[19] Arpit Gupta, Laurent Vanbever, Muhammad Shahbaz, Sean P. Donovan, Brandon Schlinker, Nick Feamster, Jennifer Rexford, Scott Shenker, Russ Clark, and Ethan Katz-Bassett. SDX: A Software Defined Internet Exchange. *SIGCOMM Comput. Commun. Rev.*, 44(4):551–562, August 2014.

[20] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving High Utilization with Software-driven WAN. SIGCOMM '13, pages 15–26, New York, NY, USA, 2013. ACM.

[21] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a Globally-deployed Software Defined WAN. SIGCOMM '13, pages 3–14, New York, NY, USA, 2013. ACM.

[22] Yuchen Jin, Colin Scott, Amogh Dhamdhere, Vasileios Giotsas, Arvind Krishnamurthy, and Scott Shenker. Stable and Practical AS Relationship Inference with ProbLink. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 581–598, Boston, MA, February 2019. USENIX Association.

[23] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the tightrope: Responsive yet stable traffic engineering. SIGCOMM '05, pages 253–264, New York, NY, USA, 2005. ACM.

[24] Eric Keller, Michael Schapira, and Jennifer Rexford. Rehoming edge links for better traffic engineering. *SIGCOMM Comput. Commun. Rev.*, 42(2):65–71, March 2012.

[25] Nandini Krishnaswamy, Mariam Kiran, Kunal Singh, and Bashir Mohammed. Data-Driven Learning to Predict WAN Network Traffic. In *Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics*, SNTA '20, pages 11–18, New York, NY, USA, 2020. Association for Computing Machinery.

[26] Harsha Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An information plane for distributed services. In *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 06)*, Seattle, WA, November 2006. USENIX Association.

[27] N. Michael and A. Tang. Halo: Hop-by-hop adaptive link-state optimal routing. *IEEE/ACM Transactions on Networking*, 23(6):1862–1875, Dec 2015.

[28] Wolfgang Mühlbauer, Anja Feldmann, Olaf Maennel, Matthew Roughan, and Steve Uhlig. Building an AS-Topology Model That Captures Route Diversity. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, page 195–206, New York, NY, USA, 2006. Association for Computing Machinery.

[29] Wolfgang Mühlbauer, Steve Uhlig, Bingjie Fu, Mickael Meulle, and Olaf Maennel. In search for an appropriate granularity to model routing policies. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, page 145–156, New York, NY, USA, 2007. Association for Computing Machinery.

[30] Usama Naseer, Luca Niccolini, Udip Pant, Alan Frindell, Ranjeeth Dasineni, and Theophilus A. Benson. Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 529–541, New York, NY, USA, 2020. Association for Computing Machinery.

[31] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. Ip geolocation databases: Unreliable? *SIGCOMM Comput. Commun. Rev.*, 41(2):53–56, apr 2011.

[32] Enric Pujol, Ingmar Poese, Johannes Zerwas, Georgios Smaragdakis, and Anja Feldmann. Steering hyper-giants' traffic at scale. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, CoNEXT '19, pages 82–95, New York, NY, USA, December 2019. Association for Computing Machinery.

[33] Bruno Quoitin, Cristel Pelsser, Olivier Bonaventure, and Steve Uhlig. A performance evaluation of BGP-based traffic engineering. *International Journal of Network Management*, 15(3):177–191, 2005.

[34] F. Schimbinschi, X. V. Nguyen, J. Bailey, C. Leckie, H. Vu, and R. Kotagiri. Traffic forecasting in complex urban networks: Leveraging big data and machine learning. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1019–1024, Oct 2015.

[35] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V. Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. Engineering Egress with Edge Fabric: Steering Oceans of Content to the World. In *SIGCOMM*, pages 418–431, New York, NY, USA, 2017. ACM.

[36] Rachee Singh, David Tench, Phillipa Gill, and Andrew McGregor. Predictroute: A network path prediction toolkit. *Proc. ACM Meas. Anal. Comput. Syst.*, 5(2), jun 2021.

[37] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2 of *INFOCOM*, pages 618–627 vol.2. IEEE, 2002.

[38] Steve Uhlig and Olivier Bonaventure. Designing BGP-based outbound traffic engineering techniques for stub ASes. *SIGCOMM Comput. Commun. Rev.*, 34(5):89–106, oct 2004.

[39] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. Cope: Traffic engineering in dynamic networks. SIGCOMM '06, pages 99–110, New York, NY, USA, 2006. ACM.

[40] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, Victor Lin, Colin Rice, Brian Rogan, Arjun Singh, Bert Tanaka, Manish Verma, Puneet Sood, Mukarram Tariq, Matt Tierney, Dzevad Trumic, Vytautas Valancius, Calvin Ying, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *SIGCOMM*, pages 432–445, New York, NY, USA, 2017. ACM.

[41] Harry Zhang. The optimality of Naive Bayes. In Valerie Barr and Zdravko Markov, editors, *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, pages 562–567. AAAI Press, 2004.

APPENDICES

These appendices contain all of our additional data, experiments and motivation for TIPSY that was too long to include in the main paper. This may serve as a reference for the interested reader.

A NAÏVE BAYES

Here we present additional learning models for ingress traffic prediction that are based on Naïve Bayes machine learning classifiers. As in the Historical Model defined in 3.3.1, for a given flow $f = \langle f_1, f_2, \ldots, f_n \rangle$, where f_k is a feature (e.g., f_1 = source AS, f_2 = source prefix, etc), the Naïve Bayes classifier (NB) [15] computes the probability that ingress(f) = l, which we write as $p(l|f)$. Using Bayes rule:

$$p(l|f) \propto p(f|l)p(l).$$

As with the Historical model, $\mathcal{L}_P^f$ is then simply the set of top-k peering links according to $p(l|f)$.

Under a strong independence assumption among the features, then $p(f|l) = \prod_{i=1}^{n} p(f_i|l)$, and we can compute each $p(f_i|l)$ from the training data by counting the observed ingress traffic and weighting it by the observed traffic volume for each feature. For example, if $B(\cdot)$ represents the total bytes seen in training for a given condition $(\cdot)$, then:

$$p(AS = 3|l = 129) = \frac{B(AS = 3, l = 129)}{B(l = 129)}$$

While the independence assumption is often violated in practice, and in our setting in particular, others have found this model to achieve surprisingly good performance in many settings [41]. It has an advantage in that it can compute predictions on flows that are not present in training, from the features of similar flows and can easily handle weighting.

For training and testing Naïve Bayes models we used Apache Spark's implementation (MLlib package). We were not able to train a NB model using source prefix as a feature (NB_{AP}), as that model exceeded Databricks memory limitations, due to massive model size (see Table 11). Using the subsets of tuples described in Table 1, we trained and evaluated two NB models: NB_A and NB_{AL} (Table 8).

Type	Name	Features
NB	NB_A	AS, Dest
	NB_{AL}	AS, Source Location, Dest

Table 8: The two Naïve Bayes models we evaluated.

We tested Naïve Bayes models and an ensemble of the Historical model with location and the Naïve Bayes with location, using older data than the used in the evaluation section: 3 weeks of training (October 1–21, 2020), and one week of testing (October 22–28, 2020). The overall accuracy is shown in table 9, and accuracy for outages is shown in table 10. While the top-3 accuracies are high, the performance of the Naïve Bayes models is inferior to the historical models, while being orders of magnitude more expensive.

Model Costs As the Historical Model, it only requires one pass on the data to train. Table 11 summarizes the runtime and memory costs for the NB models.

Model	Top 1 %	Top 2 %	Top 3 %
$Oracle_A$	66.29	86.10	91.84
$Hist_A$	63.21	83.47	89.98
NB_A	60.11	80.55	87.48
$Oracle_{AP}$	77.05	94.82	97.60
$Hist_{AP}$	73.54	92.88	96.01
$Oracle_{AL}$	75.69	94.96	98.02
$Hist_{AL}$	70.21	90.74	94.39
NB_{AL}	67.25	88.56	93.29
$Hist_{AL}/NB_{AL}$	70.85	91.65	95.47
$Hist_{AP/AL/A}$	**73.70**	**93.24**	**96.41**
$Hist_{AL/AP/A}$	71.04	91.82	95.63

Table 9: Overall prediction accuracy, with 3 weeks of training and 1 week of testing.

Model	Top 1 %	Top 2 %	Top 3 %
$Oracle_A$	57.10	80.84	86.87
$Hist_A$	34.17	51.18	66.53
NB_A	29.68	45.67	51.87
$Oracle_{AP}$	68.70	90.54	93.57
$Hist_{AP}$	30.01	51.00	71.00
$Oracle_{AL}$	68.19	90.64	94.71
$Hist_{AL}$	41.46	59.81	73.82
NB_{AL}	38.50	56.08	65.07
$Hist_{AL}/NB_{AL}$	38.97	59.08	74.74
$Hist_{AP/AL/A}$	37.48	59.14	**79.54**
$Hist_{AL/AP/A}$	**41.63**	**60.75**	75.76

Table 10: Prediction accuracy, all outages: with 3 weeks of training and 1 week of testing, only for traffic that would have arrived on a failed peering link

Model	Training	One prediction	Model size
NB_A	$O(n)$	$O(l \log l)$	$O(l \cdot (\lvert AS\rvert + \lvert DR\rvert + \lvert DT\rvert))$
NB_{AP}	$O(n)$	$O(l \log l)$	$O(l \cdot (\lvert AS\rvert + \lvert Pr\rvert + \lvert DR\rvert + \lvert DT\rvert))$
NB_{AL}	$O(n)$	$O(l \log l)$	$O(l \cdot (\lvert AS\rvert + \lvert SL\rvert + \lvert DR\rvert + \lvert DT\rvert))$

Table 11: Costs in terms of training, prediction, and memory complexity for the Naïve Bayes models. n is the number of data points; l the number of peering links; $|A|$,$|AP|$, and $|AL|$ are the number of unique combinations of the features defined in Table 1; $|AS|$,$|Pr|$,$|SL|$,$|DR|$, and $|DT|$ are the number of different ASes, prefixes, source locations, destination regions, and destination types, respectively.

For training the Naïve Bayes model, processing and memory are linear with the number of measurements (as the features are assumed to be independent). The resulting model is linear in size with the cardinality of each feature times the number of classes (all ingress peering links, over 1,000). For models with a large feature space, the size of the Naïve Bayes model actually exceeds the size of the historical model, as the number of measured feature combinations is smaller than the number of features times 1,000.

Predictions with the Naïve Bayes model are significantly more expensive than with the historical model. For each flow tuple the model computes the probability of each peering link given each feature, and then sorts the links by their probability. This can be

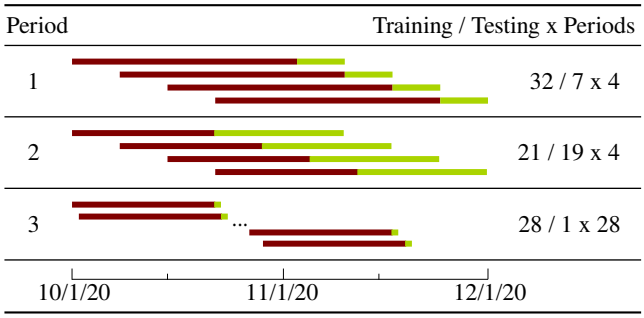

Figure 8: Time periods between October and December 2020 used for training and testing in tuning and evaluating our models. Training and testing windows are measured in days.

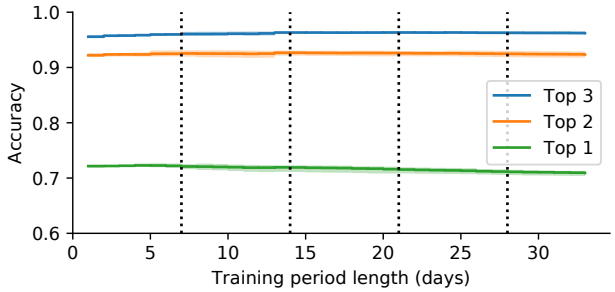

Figure 9: Accuracy of Hist$_{AL/AP/A}$ given the number of training days. Line shows average and shaded area represents minimum to maximum. The graph for Hist$_{AL}$ is similar.

done in parallel for each flow tuple of interest. As opposed to the historical model, the Naïve Bayes model allows querying for flow tuples that were not observed in the training data, as long as there was at least one tuple in the training period for each value of a feature in the desired tuple.

B TRAINING AND TESTING TIME PERIODS

We performed an empirical analysis to determine the best lengths of time for training the models. We want to balance having enough training data and avoiding stale data.

For training and testing periods, we select two months in the last half of 2020 for the purpose of our evaluation, depicted in Figure 8.

B.1 Length of training window

To identify the appropriate training window length, we fixed the testing period start date and built models using training data going back in time for varying number of days. Since the accuracy of models depends on the testing period, we selected 4 non-overlapping testing periods, and for each period built models using multiple training period lengths. The training and testing periods are period 1 in Figure 8.

Figure 9 shows the average accuracy of Hist$_{AL/AP/A}$ across the 4 non-overlapping testing periods, at varying training period lengths. The 4 different testing start days are spaced 1 week apart from

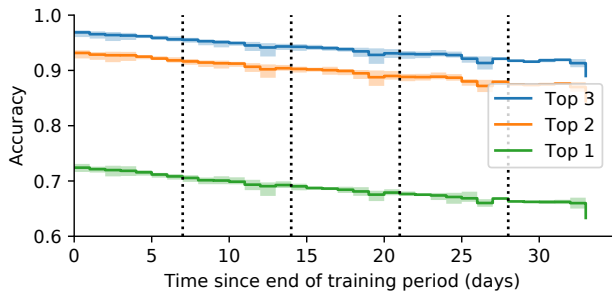

Figure 10: Daily accuracy of Hist$_{AL/AP/A}$ after training. Line shows average and shaded area represents minimum to maximum. The graph for Hist$_{AL}$ is similar.

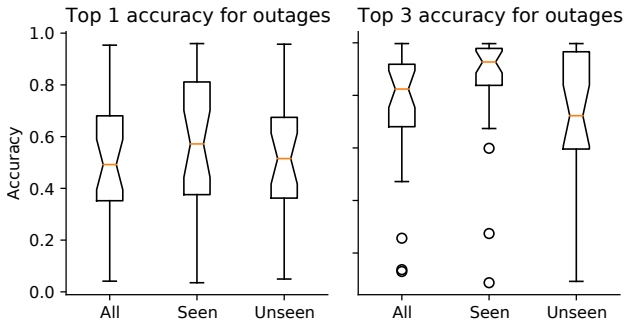

Figure 11: Accuracy of Hist$_{AL/AP/A}$ for 28 training and testing time windows. The graphs for Hist$_{AL}$ are very similar. Whiskers follow Tukey's definition.

each other, and each testing period is 7 days. *We pick 21 days as the training period for TIPSY, as it provides high accuracy for top 3 with low variability.*

B.2 Length of testing window

The routes that traffic take to reach the Azure WAN change over time, due to changes in the Internet topology, or changes in router configurations at various networks. New flows may arrive that have properties never seen before (e.g. from a new AS). We expect those changes to happen at timescales of days. For how many days is a TIPSY model valid?

In Figure 10, we use a 3-week training period, followed by testing on individual days that are progressively further out. Accuracy is calculated for non-overlapping windows of 24 hours. As before, we pick 4 different times, and show the average and the min and max. We observe an almost linear degradation of accuracy over time. The training and testing periods are period 2 in Figure 8.

We pick 7 days as an appropriate testing window. A one-week period covers commonly observed diurnal and weekly traffic patterns and does not have a significant loss in accuracy.

Algorithm 1: Identify peering links at risk

```
forall hours in the testing window do
    forall peering links A do
        forall flows that ingress on link A in that hour do
            predict which other links that flow will arrive on if
              link A has an outage
        end
    end
    sum up the bytes on each link in that hour (both actual ingress
      and outage induced ingress
end
select only those links where average utilization in any hour is
predicted to be >=70%, and would have been <70% if there was no
outage and sort the resulting list by the number of hours where the
link has average utilization >=70%
```

B.3 Sensitivity to outages in training

The impact of BGP withdrawals from seen and unseen link outages on accuracy varies greatly depending on what happened during the training window and during the testing window. To quantify this range, we trained 28 different models, each tested on 1 day of traffic, with training for the preceding 21 days, and none of these testing days overlapped. Figure 11 shows the distribution of accuracy for different outage types. The training and testing periods are period 3 in Figure 8).

C PREDICTING PEERING LINKS AT RISK

While TIPSY's main use is in mitigating ongoing congestion on peering links, the ability to predict the ingress peering link for traffic is useful in other situations as well. For instance, by leveraging TIPSY, we can identify peering links that may be at risk of overload when other peering links fail. This information is useful for capacity planning and analyzing fault tolerance of the WAN. Specifically, if a peering link A has an outage, do the flows that would have arrived on peering link A now get shifted to other peering links, where of them, some peering link B is subsequently at risk of overload? Provisioning sufficient additional capacity on link B in advance requires weeks or more of effort, and has financial implications.

We use TIPSY to identify such links at risk of overload on any other link outage, as defined in Algorithm 1. Though we pick 70% as an arbitrary threshold, we consider it to be exceedingly high because during a one-hour window with average utilization of 70%, due to the bursty nature of traffic, there will likely be a lot of queueing delays and packet drops.

In Table 12, we list four of our findings for the fourth week of October 2020 (the testing week). These examples are operationally surprising – normally, they rarely experience high utilization, however, if a particular other link has an outage, these links will experience high utilization. For the first link, if there is an outage of L1-b CPN1 in any one of 73 particular hours of the week, there will be high utilization on L1-a CPN1. Some findings are obvious in retrospect – if one link has an outage, the next closest link with the same peer will overload. However, others such as the third example are not obvious, both because the peers are different but also because the two router locations are far apart. In prioritizing which links the operator should increase capacity on, we also consider how many different affecting links cause an impact.

Router	Peer	BW	>70% hours		Affecting		
			Typical	Predicted	Router	Peer	BW
L1-a	CN1	10G	0	73	L1-b	CN1	10G
L2-a	ISP1	10G	0	28	L3-b	ISP1	10G
L4-a	CN2	20G	3	12	L5-a	EXCH1	100G
L6-a	CP1	100G	0	12	L6-b	CP1	100G

Table 12: Four actual peering links at risk of overload on individual link outage, using $Hist_{AL}$ model. "Typical >70% hours" are the number of hours in the week when utilization actually exceeded 70%, "Predicted >70% hours" are the extra number of such hours under a single peering link outage, and "Affecting" are the details of which peering link outage causes this. CN = Content Network, CP = Cloud Provider, EXCH = Peering Exchange.

Model	Top 1 %	Top 2 %	Top 3 %
$Oracle_A$	68.60	87.73	92.91
$Hist_A$	68.54	87.70	92.88
$Oracle_{AP}$	78.99	95.80	98.03
$Hist_{AP}$	78.90	95.77	98.01
$Oracle_{AL}$	78.19	96.01	98.44
$Hist_{AL}$	78.10	**95.98**	**98.42**
$Hist_{AP/AL/A}$	78.90	95.77	98.01
$Hist_{AL/AP/A}$	**78.14**	**95.98**	**98.42**

Table 13: Overall prediction accuracy, with 3 weeks of training and 1 week of testing.

Model	Top 1 %	Top 2 %	Top 3 %
$Oracle_A$	66.28	82.17	90.40
$Hist_A$	66.00	77.88	90.40
$Oracle_{AP}$	82.54	92.71	97.27
$Hist_{AP}$	**81.77**	89.15	97.17
$Oracle_{AL}$	81.68	93.14	98.07
$Hist_{AL}$	80.76	**90.70**	**98.03**
$Hist_{AP/AL/A}$	**81.77**	89.15	97.17
$Hist_{AL/AP/A}$	80.76	**90.70**	**98.03**

Table 14: Prediction accuracy, all outages: with 3 weeks of training and 1 week of testing, for traffic that would have arrived on a peering link with an outage.

D TIPSY PREDICTION ACCURACY FOR JANUARY 2021

To show results of TIPSY in another time period, Tables 13 and 14 summarize TIPSY accuracy for January 2021. The first 3 weeks of the month were used for training and the 4th week as testing (period 1). We skipped the Naïve Bayes models in this analysis. These accuracy numbers are significantly higher than those for December 2021 in Tables 4 and 5, almost on par with the relevant oracle. For this January 2021 time period, all outages in the testing

Router	Peer	BW	>70% hours		Affecting		
			Typical	Predicted	Router	Peer	BW
L7-a	ISP2	30G	1	56	L7-b	CN3	100G
L8-b	CN1	20G	9	35	L8-b	ISP3	30G
L4-a	CN2	20G	0	30	L5-a	EXCH1	100G
L3-b	ISP1	10G	0	4	L2-a	ISP1	10G

Table 15: Four actual peering links at risk of overload on individual link outage, using Hist$_{AL}$ model. "Typical >70% hours" are the number of hours in the week when utilization actually exceeded 70%, "Predicted >70% hours" are the extra number of such hours under a single peering link outage, and "Affecting" are the details of which peering link outage causes this. CN = Content Network, CP = Cloud Provider, EXCH = Peering Exchange.

period were seen in the training period. Hence this time period represents a best-case scenario.

Table 15 shows some of the links TIPSY identifies as at risk of overload for January 2021, similar to Table 12. We make a number of observations. We continue to see examples that are operationally surprising, such as the first three rows where very different peer types are involved. We also see examples such as the third row, that we saw in the previous time period, that continue to be at risk.