EPFL

Thèse n° 8182

Toward Internet Performance Transparency

Présentée le 29 août 2022

Faculté informatique et communications
Laboratoire d'architecture des réseaux
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Georgia FRAGKOULI

Acceptée sur proposition du jury

Prof. A. Wegmann, président du jury
Prof. A. Argyraki, Prof. B. A. Ford, directeurs de thèse
Prof. G. Smaragdakis, rapporteur
Prof. L. Vanbever, rapporteur
Prof. P. Thiran, rapporteur

École
polytechnique
fédérale
de Lausanne

2022

All humans by nature desire to know.
— Aristotle

To my parents, Vaggelis and Anastasia,
my sister and brother-in-law, Anna and Stelios,
and my partner, Periklis.

Acknowledgements

Foremost, I would like to thank my advisor, *Prof. Katerina Argyraki*, for her guidance, for believing in me, and for her support over the years. Katerina taught me what good research is and how to present it in the cleanest possible way. She inspires me by showing that it is possible to simultaneously be a great researcher, teacher, and person.

I also want to thank my co-advisor, *Prof. Bryan Ford*, for his guidance and support. Discussing with Bryan is always enjoyable and extremely helpful; his intellect and amazing research ideas helped me improve my work.

I am also grateful to my thesis committee members, *Prof. Georgios Smaragdakis*, *Prof. Patrick Thiran*, and *Prof. Laurent Vanbever*, for their constructive feedback on my work, and *Prof. Alain Wegmann*, for presiding over my thesis committee.

I also thank the other great professors I had the privilege to interact with at EPFL: *Prof. Anastasia Ailamaki*, for an amazing teaching-assistantship experience, and *Prof. George Candea*, for the inspiring discussions during the joint NAL-DSLAB lab meetings.

I was lucky to be surrounded by great researchers in the *NAL*, *DSLAB*, and *DEDIS labs*: *Arseniy*, *Can*, *Catalina*, *Cey*, *Cristina*, *David (x2)*, *Dimitri*, *Gaurav*, *Gaylor*, *George*, *Haoqian*, *Henry*, *Ismail*, *Jean*, *Jeff*, *Jonas*, *Kelong*, *Kirill*, *Lefteris*, *Lei*, *Linus*, *Louis-Henri*, *Ludovic*, *Luis*, *Mia*, *Muhammad*, *Nicolas*, *Noémien*, *Ovidiu*, *Pasindu*, *Pavlos*, *Philipp*, *Pierluca*, *Rishabh*, *Simone*, *Solal*, *Stevens*, *Vero*, *Yugesh*, and *Zeinab*. Thank you all for the insightful feedback and discussions that helped me improve as a researcher. Special thanks to my amazing collaborators: *Pavlos*, for always being resourceful and helping me shape parts of this thesis, and *Cristina*, *Cey*, and *Lefteris*, for challenging me to work on research topics outside my comfort zone. Finally, thanks to *Céline*, *Isabelle*, and *Sandra*, for always being available to help with administrative issues.

Many thanks to all my friends from the *DIAS lab* and beyond for making this journey much more enjoyable: *Akhil*, *Ankita*, *Antonia*, *Aunn*, *Batool*, *Christina*, *Eleni*, *Ivi*, *Panayiotis (x2)*, *Panos*, *Stella*, and *Viktor*. *Christina* has been my best friend from the moment we met. I deeply thank her for our amazing time together, excursions, and conversations. *Eleni* made life in Lausanne fun, providing me with relaxing breaks from work. Finally, *Aunn* has been a coffee buddy, and a competitive photographer during our excursions.

Acknowledgements

I am eternally grateful to *Periklis*, whose love, support, and faith in me made this thesis possible. Looking forward to our future life together!

Last but not least, I would like to thank my family—my parents, *Vaggelis* and *Anastasia*, my sister, *Anna*, and my brother-in-law, *Stelios*, for their unconditional love and support.

Lausanne, July 24, 2022 Georgia Fragkouli

Abstract

From medical support to education and remote work, our everyday lives increasingly depend on Internet performance. When users experience poor performance, however, the decentralization of the Internet allows limited visibility into *which* network is responsible. As a result, users are promised Service Level Agreements (SLAs) they cannot verify, regulators make rules they cannot enforce, and networks with competitive performance cannot reliably showcase it to attract new customers. To change this, researchers have proposed *transparency protocols*, which rely on networks reporting on their own performance. However, these proposals would be hard to adopt because i) they require substantial network resources for extracting and publishing the performance information, or ii) they require cooperative networks that honestly report their performance against their self-interests, or iii) they threaten the anonymizing capability of Tor-like networks by violating their limited visibility assumptions and introducing a new attack vector against them.

This dissertation enables network users to estimate the loss and delay of individual networks in an efficient and accurate manner, despite networks generating and controlling the performance data and potentially wanting to exaggerate their performance. It also proposes the first transparency protocol that tries to preserve the capabilities of anonymity networks.

The key to efficient and accurate performance monitoring is i) creating incentives for networks to be honest by causing dishonest networks to get into conflict with their neighbors, and ii) combining these incentives with mathematical tools that "tie together" different aspects of network performance.

The key to anonymity-preserving monitoring is the insight that users can benefit from transparency even when networks expose coarser-than-per-packet performance information, which at the same time hides sensitive communication patterns and improves anonymity.

Our thesis is that efficient and accurate Internet performance transparency is possible and that we can ease the tussle between transparency and user anonymity.

Keywords: monitoring, Internet performance transparency, verifiable network policies, incentives, anonymous communications, Tor

Résumé

De l'assistance médicale à l'éducation et au travail à distance, notre vie quotidienne dépend aujourd'hui de plus en plus de la performance d'Internet. Cependant, lorsque les utilisateurs connaissent des performances médiocres, la décentralisation de l'Internet n'offre que des informations partielles sur le réseau responsable. En conséquence, les utilisateurs se voient promettre des accords de niveau de service (Service Level Agreements – SLAs) qu'ils ne peuvent pas vérifier, les régulateurs établissent des règles qu'ils ne peuvent pas appliquer et les réseaux avec des performances compétitives ne peuvent pas les présenter de manière fiable pour attirer des nouveaux clients. Pour remédier à cette situation, des chercheurs ont proposé des *protocoles de transparence*, qui s'appuient sur les rapports des réseaux sur leurs propres performances. Cependant, ces propositions sont difficiles à adopter car i) elles nécessitent des ressources substantielles pour extraire et publier les informations de performances, ou ii) elles nécessitent des réseaux coopératifs qui rapportent honnêtement leurs performances contre leurs propres intérêts, ou iii) elles menacent la capacité des réseaux de type Tor d'anonymiser leurs utilisateurs, en violant leur proposition de visibilité limitée et en introduisant un nouveau vecteur d'attaque contre eux.

Cette thèse permet aux utilisateurs du réseau d'estimer les pertes et la latence dans des réseaux individuels de manière efficace et précise, malgré ces réseaux générant et contrôlant eux-mêmes les données de performance et voulant potentiellement les amplifier. Elle propose également le premier protocole de transparence qui tente de préserver la capacité des réseaux d'anonymiser l'identité de leurs utilisateurs.

La clé d'un contrôle efficace et précis des performances consiste à i) créer des incitations pour que les réseaux soient honnêtes en provoquant des conflits entre les réseaux malhonnêtes et leurs voisins, et ii) combiner ces incitations avec des outils mathématiques qui "relient" les différents aspects de la performance du réseau.

La clé de la surveillance qui préserve l'anonymat des utilisateurs est le fait que ces derniers peuvent bénéficier de la transparence même si les réseaux exposent des informations sur la performance plus imprécises que par-paquet, ce qui permet en même temps de masquer les schémas de communication confidentiels et d'améliorer l'anonymat.

Notre postulat est qu'une transparence efficace et précise des performances de l'Internet est possible, et que nous pouvons atténuer le conflit entre la transparence et l'anonymat des utilisateurs.

Mots-clés: surveillance, transparence des performances Internet, stratégies de réseau vérifiables, incitations, communications anonymes, Tor

Contents

List of Figures

List of Tables

1 Introduction

1.1 The Case for Transparency

Consider a user in a campus network who is giving a talk over Zoom but suddenly the video freezes because one of the multiple networks between the user and Zoom is delaying the traffic. Awareness is the first step for action, yet the user does not know which network is responsible. At the heart of this problem is the lack of Internet performance transparency [18–20, 22, 45, 65–67, 82]: When traffic is lost or delayed beyond expectation, there is no systematic, accurate way to assign responsibility to a particular network. This makes it hard for network users and/or regulators to trace problematic network behavior, or to check networks for compliance with service-level agreements (SLAs) or traffic regulations. At the same time, lack of transparency is detrimental to networks that do offer good performance, but have no systematic, accurate way of proving it. Moving toward a transparent Internet is beneficial across several axes:

Traceable performance attacks. Today, when traffic traverses multiple networks, it is hard, if not impossible, to trace a network-performance attack to its culprit. E.g., an ISP may selectively delay Bitcoin traffic, causing miners' blocks to get discarded and merchants to become victims of double-spending attacks [57]; or, equivalently, an ISP that has multiple routes (potentially of the same AS-path length) to an IP prefix may selectively route Bitcoin traffic to that prefix over the slowest/most-congested path. End-users may suspect the misbehavior but have no way of tracing it to a particular network. A transparency protocol would enable them to do so, because it would expose the delay mean and variance experienced by Bitcoin versus non-Bitcoin traffic aggregates in each network.

Verifiable good performance. Today, it is easy for a network to blame its performance problems on a neighbor, and hard, if not impossible, for the neighbor to prove its innocence. Consider, for example, the Comcast/Cogent dispute [7]: Cogent, a network operator, was transiting video traffic between Netflix and another ISP, Comcast. Comcast's customers started experiencing unexpectedly bad performance when streaming from Netflix via Cogent; the two ISPs pointed fingers at each other, and neither could prove their accusations or innocence; the dispute (and the

performance problem) went on for months. At that point, Comcast welcomed "an investigation which will allow [...] full transparency into the entire Internet backbone ecosystem" [6]. In general, transparency "is an opportunity for reliable ISPs to showcase their good performance and to distinguish themselves from the competition, which could help them attract new customers" [45].

Enforceable SLAs and regulations. Without transparency, networks cannot be checked for compliance with SLAs or traffic regulations. For example, when an ISP promises "less than 1% packet loss on all intra-domain paths," or "up to 50msec delay on all intra-domain paths" [1], or "no differentiation against video traffic," there is no systematic, reliable way to confirm that the commitment is met. Administrators typically use traceroute to estimate loss and delay on a network segment; however, the loss and delay encountered by traceroute probes may—accidentally or deliberately—differ arbitrarily from those encountered by other traffic, so nothing can be said about the accuracy of such measurements.

We are *not* arguing for more SLAs or traffic regulations; but for the SLAs that are already in place, there should be a way to verify them. If ISPs choose to state loss and delay bounds in their SLAs, they must believe that their customers care about loss and delay. If the European Union parliament has incurred the cost to produce neutrality regulations [8], its members must believe that their citizens care about network neutrality. Without any technical means for identifying where packet loss, delay, or traffic differentiation occurs, SLAs and regulations only serve to create the illusion of commitments that cannot be checked.

Principled de-regulation. Transparency is necessary not only to enforce regulation, but also to enable de-regulation to work as intended. During the Trump administration, the FCC abolished neutrality regulations[1] and required, instead, "internet service providers to be transparent about their practices so that consumers can buy the service plan that's best for them and entrepreneurs and other small businesses can have the technical information they need to innovate" [11, 15]. So, the very document that abolished neutrality regulation argued for transparency in its place.

1.2 Goals and Challenges

To improve Internet performance transparency, researchers have proposed *transparency protocols* [18–20, 22, 65–67, 82], where networks report on their own performance.

For example, consider a packet flowing through a sequence of two networks (Fig. 1.1). Network users know that it takes 100msec for the packet to go from the entry of N_1 to the exit of N_2. However, users do not know the breakdown of the packet delay along the path: out of the 100msec, 40msec are spent within each network, and 20msec over the inter-domain link. To make each network's delay transparent to users, we put in place a transparency protocol in which each network reports when each packet enters and exits.

[1] The Biden administration is expected to revive them. Also, the State of California enacted its own neutrality regulations in the meantime.

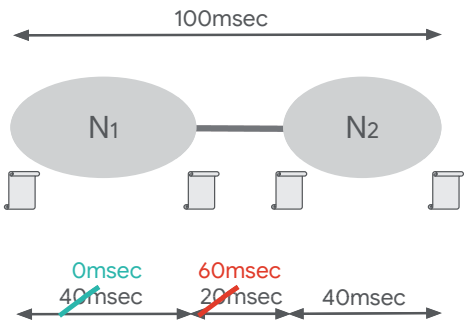


Figure 1.1 – Toy example: A packet traverses a sequence of two networks that have deployed a transparency protocol. The networks report when the packet enters/exits them and users can track packet loss and delay to a particular network. Each segment is labeled with its actual packet delay (in gray) and the exaggerated one (in green or red), which is the result of N_1 altering its exit report to claim zero internal delay.

However, improving transparency is challenging because of the business model and scale of the Internet, the diverse performance interests of users, and the limited-visibility assumptions that anonymity frameworks atop the Internet rely on. More specifically:

- **Accuracy despite network self-interests.** What if N_1 delays the packet but alters its exit report to claim zero internal delay?

  Users want to know the networks' true performance, but it is the networks that generate and control the performance data and may want to exaggerate their performance (i.e., hide the fact that they drop/delay traffic) to increase their revenue. Thus, there is a conflict of interest between those who produce and those who want to access the data.

- **Accuracy for flexible user interests.** What if N_1 delays the packet and blames it on N_2, but users do not care about what happened to individual packets?

  Prior work [18, 19] achieves accurate loss and delay statistics through i) networks reporting on every single packet or TCP flow and ii) incentive structures that rely on the *threat-of-conflict* notion and create conflicts between networks under dishonesty. These incentives, however, only hold if users care about the fate of individual packets/TCP flows. And yet, not all packets equally affect user experience: a delayed Bitcoin packet carrying expensive transactions would arguably result in users complaining to their ISPs; a single delayed Netflix packet less so. Thus, there is a mismatch between what the incentive structures assume and what users actually care about.

- **Accuracy despite monitoring efficiency.** Transparency involves networks that extract performance information from the data plane, but this procedure must be lightweight in terms of network resources if we want transparency solutions to be deployed in the core of the Internet that observes Terabits of traffic per second. The straightforward solution seems to be sampling: instead of networks reporting on every packet they observe, they could

report on a small packet sample (e.g., 5%). The flip side of sampling is reduced visibility which networks may exploit to exaggerate their performance. Hence, transparency has always involved a trade-off between accuracy and efficiency.

- **Accuracy despite user anonymity.** Transparency introduces an attack vector against user anonymity: Performance reports produced at strategic network points reveal when certain traffic appeared at a certain network point. This violates the assumptions of popular low-latency anonymity networks, like Tor, which rely on limited transparency to deliver their anonymity goals.

1.3 Contributions

This thesis makes the following contributions:

(1) We propose a new transparency protocol that makes it possible to accurately estimate each network's loss average, delay average, and jitter, based on measurements of a small sample of the networks' traffic. Our protocol relies on the following key ideas:

- **Alignment with user interests.** Prior work assumed that the incentives created by users would adapt to the incentives required by transparency protocols, i.e., that users would care about the fate of individual packets. However, applications often handle individual packet drops/delays without impact on user experience. This breaks the incentive structure of existing transparency protocols and challenges accurately computing aggregate metrics that interest users. We take the opposite, more natural approach and design a transparency protocol that adapts to user interests. To this end, we introduce the concept of user-defined *traffic aggregates* and create honesty incentives around aggregates. For example, users may define traffic aggregates consisting of individual Bitcoin packets, but they may also define coarser aggregates containing Netflix packets exchanged within the last hour.
- **Incentives for honesty through conflict with neighbors.** We create threat-of-conflict incentives for networks to honestly report per-aggregate loss and delay averages: we propose a new, intuitive definition of loss and delay averages such that if a network claims lower-than-true loss/delay, it necessarily pushes loss/delay blame to a neighboring network and risks entering conflict. Entering a conflict penalizes the dishonest network and acts as an incentive for honest reporting.
- **Incentives for honesty through inter-dependent network performance aspects.** Conflict with neighbors alone is not enough to incentivize networks to honestly report their jitter (because jitter is not additive). For that, we identify an intrinsic interplay between per-aggregate delay averages and jitter. Specifically, if a network exposes a set of traffic aggregates to the same network conditions (e.g., to the same sequence of links/buffers), its delay averages w.r.t. these aggregates cannot vary arbitrarily but must

meet certain simple, well-defined mathematical constraints. From these constraints (and the honestly reported per-aggregate delay averages), users estimate each network's jitter.

(2) We perform the first analysis of a transparency protocol from an anonymity perspective and show that: i) A transparency protocol indeed risks de-anonymizing user flows, and this risk increases over time, as the protocol produces more reports. ii) Adding noise to the performance reports to ensure differential privacy would not work: one would need to add so much noise that the performance reports would become useless. Instead, we build on the insight that it is possible for performance reports to be coarse enough to hide sensitive information about individual users, yet detailed enough to benefit network applications. For example, when a user downloads a movie or a kernel distribution, that typically takes several seconds or even minutes; hence, the user cares about the network's aggregate performance over several seconds or minutes (not about what happens to every single packet). Based on this insight, we propose an algorithm that continuously adapts the time granularity of the performance reports, i.e., makes them as coarse as necessary, to hide the individual flow patterns that stand out the most, thus improving anonymity.

The contributions of this thesis can be summarized as follows:

We show that it is possible to improve Internet performance transparency—accurately estimate network loss, delay, and jitter—without requiring more than a few percentage points increase in network resources and without having to trust the measurements that networks provide. We also show that it is possible to ease the tussle between transparency and user anonymity.

1.4 Thesis Outline

The rest of this thesis is organized as follows:

- Chapter 2 presents the necessary background this thesis builds on.
- Chapter 3 shows how a transparency protocol can adapt to user interests and achieve accurate loss and delay average estimates despite network self-interests.
- Chapter 4 shows how to accurately estimate jitter despite threat-of-conflict incentives not being enough for honestly reporting jitter. For that, we introduce a two-step approach that builds on the honestly reported per-aggregate delay averages and a new technique for verifying whether a network has exposed a set of traffic aggregates to the same network conditions.
- Chapter 5 shows that transparency is at odds with user anonymity and proposes adapting traffic reports to hide sensitive user communications patterns.
- Chapter 6 concludes the thesis and discusses future research directions.

2 Background

Assuming traffic flowing through a chain of *networks*, such as datacenters, enterprises, campuses, home networks, Autonomous Systems (ASes), or Internet Service Providers (ISPs), the goal is to enable a third party, such as network users or regulators, to monitor the performance of each network. This chapter provides the necessary background on the existing tools, techniques and the challenges they face.

2.1 Monitoring from the Edge

Past work [28, 30, 37, 38, 53, 56, 64, 75, 83, 84] has built techniques and tools that estimate the performance of network segments (single network links or entire networks) from end-to-end (E2E) measurements, potentially crowd-sourced to large user bases through mobile applications. Because these tools entirely rely on observations at the edge of a network path, i.e., without directly monitoring each link/network, they require minimal network cooperation, deployment effort, and overhead. They also allow accurate detection of performance issues end-to-end, but depending on the techniques used, they may not be able to localize performance issues to specific links/networks. And while it is sometimes reasonable to assume that a given performance issue (e.g., traffic differentiation) on an E2E path is caused by a given ISP (e.g., the access ISP) [28], that is is not always the case and leads to false accusations that are detrimental to innocent ISPs.

Network performance tomography. Tomography techniques [30, 37, 38, 56, 64, 84] infer the performance (e.g., loss rate, latency, congestion, traffic differentiation) of network segments despite having access only to E2E measurements and the topology of the vantage points. In a nutshell, tomography exploits correlations in E2E measurements and uses the topology to build a system of equations, where vantage points measure E2E performance and network-segment performance is inferred by solving the system of equations. However, the capabilities of tomography techniques greatly depend on creating the "right" topology of vantage points: to reason about a network segment, tomography needs vantage points carefully placed at different paths that intersect only at that segment. It is challenging to create topologies that meet this

constraint, especially in our context where only network users (vantage points) on the same path may be interested in the performance of a given network at a given time, hence willing to spend the compute and bandwidth resources required for the measurements.

2.2 Standard Networking Tools

Users could leverage standard networking tools, such as ping [54] and traceroute [55], to estimate loss and delay on a network segment. Although the details differ, these tools work by having the source send probe packets to a destination; upon hitting on-path routers or the destination, probe packets trigger packet replies to the source. The source estimates loss rate and round-trip time (RTT) between any two nodes on the probes' path by combining the departure/arrival times of probe packets and the corresponding replies. Thus, the effectiveness of probing tools fundamentally relies on cooperative routers that respond to probe packets. However, at times of network congestion or outages resulting from misconfigurations [72], organizations may delay reporting [76] or may not even want to admit to facing performance issues, as this often results in customer dissatisfaction and monetary loss. We can expect that networks will block the probing tools or treat probe packets preferentially to exaggerate network performance. As a result, standard monitoring solutions may fail to report performance metrics or fail to reflect the actual performance experienced by the bulk of the traffic.

2.3 Transparency Protocols

Researchers have proposed *transparency protocols* [18–20, 22, 65–67, 82] that localize performance issues to networks by directly monitoring the performance of individual networks. Networks participating in a transparency protocol deploy and control special logic, called *witnesses*, at their entry and exit points, e.g., at the linecards of border routers or within the firmware of an enterprise/home gateway. Witnesses observe traffic and continuously produce performance reports, called *statements*, on the traffic they observe. Periodically, networks send the performance reports to regulatory entities, called *monitors*, who assess and make available the performance of each network to network users. A monitor may be run by any organization that provides Internet services (e.g., ICANN [12]), a regulator (e.g., FCC [9] or BEREC [2]), a user collective, or a subset of the participating networks. Multiple monitors may operate simultaneously and independently from each other, possibly generating different performance conclusions. Users follow the monitor they trust.

While conceptually a straightforward solution, transparency protocols may not always allow a monitor to accurately assess network behavior: because networks control the witnesses and are essentially self-reporting, they can arbitrarily modify the generated reports, either unintentionally due to software bugs or intentionally to hide performance issues from customers and avoid monetary loss. So, transparency protocols must cope with dishonest networks to enable a monitor to accurately assess network performance.

2.3.1 Per-packet Reports

To achieve accurate monitoring, transparency protocols need to impose some penalty on networks that dishonestly declare their performance. However, a monitor cannot directly penalize dishonest networks because it does not know which network is dishonest. This section reviews the mechanism that transparency protocols use to achieve accurate monitoring despite network self-interests, showing the limitations of applying the mechanism for every packet or TCP flow.

A monitor indirectly penalizes dishonest networks by making them conflict with neighboring networks. Early work [18, 19] relies on the *threat-of-conflict* notion regarding individual packets or TCP flows: Because networks report on every packet or TCP flow, a network cannot hide that it lost or delayed a packet/flow; it can only blame the loss/delay on its neighbors or inter-domain links. However, if a network blames a neighbor or inter-domain in this way, the neighbor observes and can dispute the malicious claim, putting the two networks in conflict with each other. Thus, networks have an incentive to honestly report per-packet or per-TCP-flow loss and delay.

This design assumes that network users care about the fate of *individual* packets or TCP flows and that networks care (and would enter conflict) to prove that they did not lose or delay individual packets/flows. Assuming that conflicts happen over individual packets is a crucial assumption on which the performance conclusions of a monitor greatly depend: If networks have an incentive to accurately report the loss/delay of every packet, then a monitor can accurately compute *any* aggregate performance metric (e.g., loss rates, delay averages, jitter) as a function of accurate per-packet loss/delay. In contrast, if networks have no incentive to honestly report the loss/delay of every packet, existing transparency protocols are silent on what performance metrics a monitor can still accurately compute.

While a crucial assumption, it is not always reasonable to assume that conflicts happen over individual packets. For example, missing an individual packet has almost no impact on real-time video streaming applications [59, 78]. So, users may not even notice, let alone complain to their ISPs about losing the packet. But without any bad consequences of losing the packet, even if the ISP that lost that packet blames it on a neighbor, the neighbor has no reason to conflict with the dishonest ISP. Hence, there is no incentive for the ISP that lost the packet to report so honestly. In contrast, what many applications consider a significant drop in quality of service is a drop in the performance of *aggregates* of packets, such as back-to-back burst losses [59] and packet loss above a certain level [63]. So, there is a mismatch between the incentives created by user interests and the incentives required by existing transparency protocols, which causes a monitor to make arbitrarily inaccurate conclusions.

Per-packet or per-TCP-flow approaches require significant bandwidth and data-path state to maintain and extract statements from the data plane. However, in modern network devices, there is only modest bandwidth between the data plane and the local control plane (whether that is a traditional supervisor engine, or a minimal OpenFlow agent that acts as an intermediary to an external controller). Increasing this bandwidth to continuously export per-packet statements on

billions of packets per second would require a significant shift in hardware design. Per-TCP-flow statements are cheaper, but they still require per-TCP-flow state on the data path (counters and timestamps). Moreover, they make networks vulnerable to denial of service, where an attacker sends single-packet flows, effectively causing witnesses to maintain and emit per-packet statements.

Summary. Per-packet or per-TCP-flow reports introduce significant bandwidth overhead and there are practical scenarios where conflicts over individual packets/flows do not happen, resulting in inaccurate monitor conclusions.

2.3.2 Sampled Reports

Prior work [20, 65, 66, 81] reduces overhead by moving away from networks reporting on every packet or TCP flow; instead, networks generate reports only for a small subset of the packets. However, limiting a monitor to only a subset of the traffic introduces attack vectors for networks to exaggerate their performance without facing any consequences. This section reviews sampling-based transparency protocols and their mechanisms to mitigate attack vectors, showing that sampling-based approaches still require that conflicts happen over individual sampled packets.

Random sampling. A straightforward solution that reduces the bandwidth overhead is random sampling, where networks report on a small subset of the packets they observe (e.g., 5%) [20, 81]. The premise is that random sampling provides a monitor with a configurable, representative sample of the overall packets. Hence, networks control the resources spent on monitoring, and a monitor estimates network performance with a small and known error.

Random sampling, however, allows for sources of inaccuracy beyond the pure statistical error:

First, if networks are allowed to report on different subsets of packets, networks have an incentive to launch *collusion attacks* [20, 65] and exaggerate their performance: Each network includes in the sample the packets that it treated well; also, different networks pick non-overlapping packet subsets. Without a method for a monitor to check the representativeness of each subset and without common packets over which conflicts happen, networks exaggerate their performance without risking conflict.

Second, if networks know which packets to sample at forwarding time, they can launch *prioritization attacks* [20, 65, 66, 81] and treat the sampled packets better than the rest, e.g., by assigning them to high-priority queues or less-congested paths. As a result, networks honestly report the performance of the sampled packets, but that is not representative of the performance of the non-sampled bulk of the traffic.

Consistent sampling. To prevent collusion attacks, transparency protocols build on *trajectory sampling* [29] and replace random with *consistent sampling* [20]. At a high level, consistent sampling removes the incentive for neighboring networks to collude by making networks report

on the *same* subset of packets. Specifically, networks apply a hash function on each packet's immutable content and sample the packet if the outcome exceeds a configurable value. Because of the randomization properties of the hash function, the sampling process resembles random sampling. At the same time, because of the determinism of the hash function, networks report on the same packet sample (modulo loss), so consistent sampling allows for threat-of-conflict incentives to apply to every sampled packet.

Consistent sampling, however, faces two challenges:

First, consistent sampling alone cannot prevent prioritization attacks. But without resistance to preferential treatment of sampled packets, threat-of-conflict incentives become irrelevant: If networks can prioritize sampled packets, networks have already succeeded in exaggerating their performance and have no reason to make inconsistent performance claims with each other and risk conflict.

Second, even if prioritization attacks were not a problem, consistent sampling still relies on the strict assumption that conflicts happen and honesty incentives hold for individual sampled packets. Without this assumption, per-packet reports do not allow a monitor to accurately estimate network jitter, because jitter is not an additive metric, and one cannot build threat-of-conflict incentives for reporting it honestly [65]. More importantly, existing sampling-based transparency protocols cannot even reason about delay mean, because when delay mean is estimated from samples, the confidence interval is a function of jitter.

Delayed sampling. To mitigate prioritization attacks, prior work has contributed *delayed sampling* [20, 65, 66, 81]. In a nutshell, delayed sampling is akin to a commit-and-reveal protocol: during the commit phase, networks must forward packets without knowing which packets will be sampled; it is only during the subsequent reveal phase that networks learn the sampling fate of packets. Since by the time the reveal phase happens the packets are already forwarded, networks cannot forward/treat them preferentially. Under the covers, delayed sampling works by networks keeping temporary state on all packets and applying consistent sampling twice: once to pick a small number of "disclosure packets"; these seed another consistent sampling process that determines which of the previously collected state to keep (i.e., which packets to sample) and which to discard.

Delayed sampling is secure against exploits of the sampling procedure, but this is not enough to ensure honest reporting: once networks have learned the sampled packets, networks can arbitrarily modify the sampled performance reports before sending them to a monitor unless there is some penalty. To penalize dishonesty, existing sampling-based protocols rely on a strict assumption: that conflicts (i.e., indirect penalization) happen over individual sampled packets and not over aggregates of sampled packets that interest users.

Summary. Existing transparency protocols achieve efficient monitoring through sampling and make it secure against exploits of the sampling procedure. However, they still fail to achieve accuracy in the general case, where users do not care about the fate of individual packets. In

this thesis, we build on top of secure delayed sampling, but contrary to previous approaches, we provide a transparency protocol capable of incentivizing networks to be honest even for performance metrics that are not subject to threat-of-conflict incentives in the general case of coarse-grained user interests.

3 Split-responsibility for Verifiable, User-based Average Metrics

Average performance metrics like average latency and packet delivery rate are core to measuring the quality of service of applications and the effectiveness of Distributed Denial of Service (DDoS) defenses. However, the decentralization of the Internet does not allow network users to assess and localize performance issues to individual networks. To change this, existing transparency proposals rely on network self-reports but incentivize honest reporting through threat-of-conflict incentives that are assumed to apply to *individual* packets. Under this assumption, networks have an incentive to honestly report the performance of individual packets, which greatly simplifies the incentive structure and enables accurate computation of any aggregate performance metric. But the incentive structure is then "fixed" and cannot adapt to evolving user interests and application requirements that are often not concerned with individual dropped/delayed packets. The resulting mismatch between the incentives created by user interests and the incentives required by existing transparency protocols significantly reduces the accuracy of the computed averages.

In this chapter, we leverage conflicts to create incentives for honesty not only around individual packets but also around aggregates of packets that interest users (e.g., traffic from certain video providers, or certain gaming platforms). We propose simple but new loss and delay average metrics that hold each network responsible for its internal loss/delay and part of the loss/delay experienced on each of its inter-domain links. Thus, if a network claims lower-than-true loss/delay, it necessarily pushes loss/delay blame to its neighbor and risks entering conflict. Entering a conflict indirectly penalizes the dishonest network and acts as an incentive for honest reporting.

3.1 Overview

3.1.1 Participants

Aether, our performance-transparency protocol, involves the following entities:

A **network** is a contiguous entity managed by a single administrative authority, e.g., a datacenter, enterprise, campus, or home network; an Autonomous System (AS); or an ISP. When we say

"network" without further qualification, we refer to a network that has deployed Aether.

A **witness** is a data-plane component that continuously creates *statements* (see below) on the traffic it observes. Each network deploys a witness at each entry and exit point, e.g., at the linecards of its border routers or within the firmware of an enterprise/home gateway (see App. §B.1.1 for a discussion of where exactly witnesses are deployed and how that affects transparency).

Witnesses that belong to the same network synchronize their clocks at a granularity of a few milliseconds, while two witnesses at opposite ends of the same inter-domain link synchronize their clocks at a granularity of a few microseconds, e.g., using the algorithm in [52] (see App. §B.1.2 for a discussion on clock drift).

A **witness pair** is a pair of witnesses deployed, respectively, at an entry and an exit point of the same network. Hence, it defines one or more (in case of internal load-balancing) internal paths through the network.

A **witness manager** is a control-plane component that periodically collects statements from the network's witnesses. Each network deploys one.

Witnesses create **statements** as in prior work [20, 66]: Each witness samples packets and creates a statement per sampled packet. A statement consists of: a packet identifier (e.g., a hash of the packet's immutable content); the time when the witness observed the packet; the packet's source and destination IP prefixes and port numbers (the latter if the transport-layer header is unencrypted).

Witnesses perform *consistent sampling* [20]: a packet is sampled either by all the witnesses that observe it, or by none. This is possible using hash-based sampling [29], which applies a hash function on each packet's immutable content and samples the packet if the outcome exceeds a configurable value. Hence, when we refer to a "sampled packet," we do not specify by which witness this packet was sampled—we imply that the packet was sampled by all the witnesses that observed it.

Further, witnesses perform *delayed sampling* [20, 66], which prevents witnesses (and networks in general) from treating the sampled packets preferentially (§3.2, No clean-Diesel measurements).

A **monitor** is a system that defines *traffic aggregates* (§3.5) of interest to the network users, collects statements and other relevant information from the witness managers, and estimates the performance of the participating networks. A monitor may be run by any organization that provides Internet services (e.g., ICANN [12]), a regulator (e.g., FCC [9] or BEREC [2]), a user collective, or a subset of the participating networks. For simplicity, we will assume one monitor, but, in principle, there may exist several, operating independently from each other.

3.1.2 Threat Model and Goal

Witnesses can be "honest" or "lying":

An **honest witness** is one that emits correct statements, i.e., prepares each statement using the algorithm provided by Aether for this purpose.

A **lying witness** is one that: suppresses a statement, i.e., pretends that it never received traffic that it actually did receive and drop; emits a superfluous statement, i.e., pretends that it delivered traffic that it actually dropped; modifies a statement, i.e., pretends that it treated certain traffic differently than it actually did.

When a network's witnesses lie, the monitor may compute an "exaggerated performance" for that network. More precisely: Suppose the monitor computes network N's performance (w.r.t. some well-defined performance metric). Suppose that: if all the witnesses were honest, the monitor would compute N's performance as v. We say that N's performance is "exaggerated by δ" if the monitor computes N's performance as $v' = v - \delta$. Our definition is relative to what would result from honest reporting (as opposed to ground truth), because the monitor estimates performance based on sampling, hence its estimate may differ from the ground truth, even if all witnesses are honest; we want the definition of "exaggerated performance" to capture the effect of lying witnesses excluding sampling error.

Our goal is to design the statements created by the witnesses and the performance metrics computed by the monitor such that: (1) The participating networks incur small bandwidth overhead. (2) The monitor either computes the target performance metrics with well-defined accuracy, or declares that it cannot, because of lying tracked down to a pair of neighboring witnesses/networks.

We do not assume anything about the monitor's honesty. It is, of course, plausible that a monitor publishes performance metrics that are different from the ones it computes. This is why it makes sense for several independent monitors to operate at the same time and let network users follow the one they trust.

3.2 Design Principles

The following principles guided our design:

Neighbor-dependent metrics. A network's statements must affect both its own and its neighbors' performance metrics, in order to incentivize networks to expose their neighbors' lying. For example, consider a network N, a traffic aggregate A, and the following, naïve performance metric: N's internal loss rate with respect to A. This performance metric is computed solely from N's statements. Hence, N can lie and pretend that the packets it lost internally either never entered or always exited its network, essentially blaming its internal loss on its inter-domain links.

As long as the loss or delay that occurs on inter-domain links is not taken into account when computing network performance, networks have an incentive to independently minimize their perceived internal performance, and no incentive to expose each other's lies.

Conflict as a consequence of lying. Lying must have consequences, to incentivize networks to be honest. The monitor cannot impose direct penalties to dishonest networks, e.g., assign bad ratings or fine them, because it does not know which networks are dishonest. Instead, we rely on the conflict that happens when networks do not agree with each other's statements. For example: networks N_1 and N_2 exchange traffic over a well-provisioned inter-domain link that normally incurs no packet loss; N_1 claims that it delivered certain traffic to N_2, while N_2 claims to never have received it, implying that the traffic was lost on the link between them; the two networks may accept this, or one of them may decide that it does not make sense (either the link is malfunctioning, or the other network is lying) and declare a conflict. Prior work [18–20, 66] also relied on conflict, but regarding individual packets or TCP flows; we rely on conflict regarding larger traffic aggregates that are determined by user interests (see below). We assume that a network would *normally* want to avoid conflict with a neighbor, because that would damage their business relationship. Of course, as history [4] may suggest, two neighboring networks may still determine that their best course of action is conflict. A transparency protocol cannot prevent that, but it can make the conflict clearer, more explicit. E.g., the Comcast/Cogent dispute consisted of vague accusations about illegal practices; a transparency protocol would immediately disclose each network's performance w.r.t. Netflix traffic addressed to Comcast's customers, and would show that that traffic suffered at the inter-domain link between the two networks; the two networks would then need to explain why their inter-domain link was not functioning as expected. So, transparency cannot prevent conflict, but it can force the conflicting parties to make explicit statements. We expect that networks are less likely to lie when the lie must be explicit.

Alignment with user interests. Users must care about the traffic aggregates and performance metrics defined by the transparency protocol, otherwise the conflicts that we rely on may not happen. For example, suppose network N_1 drops some traffic and tries to blame the loss on N_2; if users don't care about the loss of this traffic, N_2 may not care either that it is being unfairly blamed for it, and there may be no conflict—and no bad consequence for N_1's lying. Given that user interests evolve, we should not hardcode specific traffic aggregates or performance metrics into the data plane, i.e., the witnesses should be agnostic w.r.t. traffic aggregates and metrics.

No "clean-Diesel" measurements. Networks must not be able to manipulate the transparency protocol by treating certain traffic preferentially. This precludes traditional packet sampling: if a network reports on a small sample of the packets it observes, and its performance is reconstructed based on these samples, the network can treat the sampled packets preferentially. To avoid this, we rely on delayed sampling [20, 65, 66, 81]: each witness keeps temporary state on all packets and uses hash-based sampling to pick a small number of "disclosure packets"; these seed another hash-based sampling process that determines which of the previously collected state to keep (i.e., which packets to sample) and which to discard. Hence, when observing a packet, a witness does not know whether that packet will be sampled or not and cannot treat it preferentially.

3.3 Split-responsibility Packet Loss and Delay

Monitoring each network's loss and delay for individual packets is a fundamental building block for estimating any performance metric. Network self-interests, however, challenge the accuracy of the performance reports. For example: two users exchange a packet through networks N_1, N_2 and their inter-domain link. The inter-domain link is *normally* not congested, yet the users observe unacceptably high end-to-end delay for the packet. N_1 claims that as soon as it received the packet, it delivered it to N_2, and N_2 makes a similar claim. Effectively, the networks push the responsibility for the "orphan delay" to the link between them, and the monitor wrongly concludes that it is the inter-domain link that delayed the packet.

Externalizability. Prior work [18] observes the *externalizability* of packet loss and delay, but externalizability by itself is not enough to incentivize accuracy: Externalizability simply states that someone must take the responsibility for the orphan loss/delay. However, as long as this someone is the inter-domain link, networks have an incentive to independently minimize their perceived internal performance by offloading the performance blame on the inter-domain link, and no incentive to expose each other's lies.

Neighbor-dependent metrics. We observe that a network's statements must affect both its own and its neighbors' performance metrics, in order to incentivize networks to expose their neighbors' lying. In Aether, we leverage this observation by holding each network responsible for its internal loss/delay plus half of the loss/delay experienced on each of its inter-domain links. In particular: Given a packet p, observed by a sequence of networks $\langle N_{i-1}, N_i, N_{i+1}\rangle$, the monitor computes N_i's *split-responsibility loss w.r.t. packet* p as:

$$l_i(p) \triangleq l + \frac{1}{2}\left(l^- + l^+\right), \tag{3.1}$$

where l, l^-, l^+ denote whether p was lost inside, on the inter-domain link before or after N_i. The monitor computes N_i's *split-responsibility delay w.r.t. packet* p in a similar way:

$$d_i(p) \triangleq d + \frac{1}{2}\left(d^- + d^+\right), \tag{3.2}$$

where d is the delay experienced by p inside N_i, while d^- (resp. d^+) is the delay experienced by p on the inter-domain link before (resp. after) N_i. The monitor computes N_i's loss and delay w.r.t. p from the packet identifiers and timestamps of N_{i-1}, N_i, and N_{i+1}'s witness statements.

Conflict as a consequence of lying. A network can lie about its loss/delay w.r.t. a packet p only in one way: it can falsely blame some of its internal loss/delay on one of its inter-domain links. This leads to the following:

Lemma 3.1 *Consider a packet p observed by a sequence of networks $\langle N_i,\ i \in \mathbb{N}\rangle$. If the monitor computes each network's loss/delay w.r.t. p, and N_i's loss/delay is exaggerated by δ_i, then $\sum_i \delta_i = 0$.*

The proof (in §3.6.1) relies on the additive property of the split-responsibility packet loss and delay. Lemma 3.1 also holds for sampled packets: because of consistent sampling, a packet is sampled by either all or none of the witnesses on its path; because of delayed disclosure, networks cannot treat sampled packets preferentially. As a result, when a packet incurs a certain amount of loss or delay, a witness cannot hide that loss/delay; it can only influence whether the loss/delay appears to have happened before or after it.

For example, suppose N_i falsely blames 2δ of its internal delay w.r.t. p to the inter-domain link with neighbor N_{i+1}. If N_{i+1} disputes, then there's conflict, and the monitor does not compute the performance of any network w.r.t. p. If N_{i+1} does nothing, then (because of the split-responsibility delays) N_i's delay is exaggerated by δ, while N_{i+1}'s delay is exaggerated by $-\delta$ (penalized by δ). If N_{i+1} pushes the false delay blame downstream to N_{i+2}—and N_{i+2} does not dispute—then N_i's delay is exaggerated by δ, N_{i+1}'s by 0, and N_{i+2}'s by $-\delta$.

Nevertheless, pushing the blame downstream comes with consequences; it can recoil upon the blamer. In the above example, suppose that not only N_{i+1}, but also N_{i+2} and all subsequent networks push the blame down to a certain network N_k that is not willing to push the blame further to N_{k+1} or is simply the destination network—hence, it conflicts with its predecessor network N_{k-1}. Similarly, N_{k-1} will now have an incentive to conflict with N_{k-2} that had pushed the blame to it, and so on. In the end, this "cascade effect" of conflicts will bounce back to N_i, which arguably gives it an incentive not to push the blame in the first place.

So: given Lemma 3.1, network N_i chooses to be honest w.r.t. packet p under the following conditions: (1) N_i prefers revealing its true performance w.r.t. p to entering conflict with a neighbor. (2) N_i's neighbors prefer entering conflict with N_i to accepting false blame w.r.t. p at their own expense or pushing it to another neighbor.

Summary. Aether forces networks to either be honest; or tell explicit lies that affect their neighbors and may lead to conflict. Whether this is sufficient for honesty depends on the health of the market. For example, a powerful player may always choose to lie knowing that its neighbors will avoid conflict with it at any cost; this scenario, however, implies an unhealthy market, e.g., a monopoly, where the powerful player is the only eyeball ISP available in a significant geographic area. We believe that there exists no transparency protocol that can incentivize such powerful players to be honest.

3.4 Adapting to User Interests

Applications often handle individual packet drops/delays without impact on user experience. For example, video streaming applications can mask packet loss from the user through error control and concealment techniques [35] that recover important missing data in the bitstream. More importantly, "there is no real need to recover all missing packets" [35]. As a result, falsifying the fate of individual packets is unlikely to negatively impact ISPs and lead to conflicts. This breaks

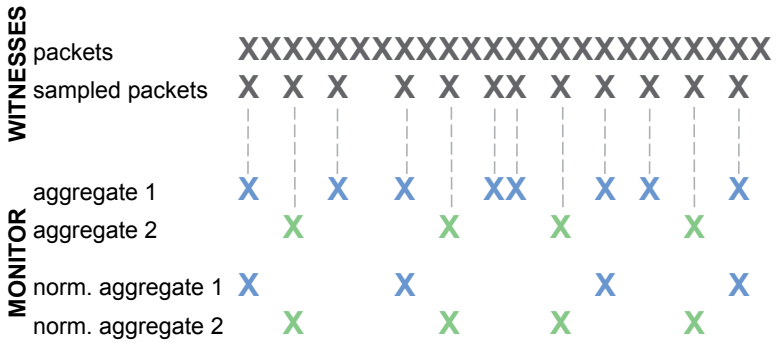


Figure 3.1 – Traffic units used in transparency: Witnesses observe and sample packets. The monitor defines traffic aggregates and maps each statement (hence each sampled packet) to an aggregate.

the incentive structure of existing transparency protocols and challenges accurately computing aggregate metrics that interest users.

We introduce the concept of user-defined *traffic aggregates* and create honesty incentives around them. Traffic aggregates capture application requirements and user interests (§3.2, Alignment with user interests) over the performance of aggregates of packets, such as back-to-back burst losses [59] and packet loss above a certain level [63]. Essentially, traffic aggregates are the basis for decoupling honestly reporting aggregate metrics from honestly reporting the performance of individual packets. Given that user interests evolve, we should not hardcode specific traffic aggregates or performance metrics into the data plane. So, networks continue reporting on sampled packets, but we create honesty incentives around the performance of user-defined aggregates of packets. More specifically:

A **traffic aggregate** is a set of packets with the same source and destination IP-address prefixes, which traverse the same sequence of witnesses.

The monitor defines traffic aggregates based on user interests. For example, currently, two performance-sensitive services that network users care for are video streaming and online gaming. To keep track of networks' performance w.r.t. these services, the monitor would define traffic aggregates of the type "traffic between IP prefixes X and Y," where X (resp. Y) belongs to a video provider or gaming platform, and Y (resp. X) belongs to an eyeball ISP. Moreover, if traffic from a given application, e.g., Bitcoin, experiences bad performance along certain network paths [57], the monitor would define traffic aggregates of the type "Bitcoin traffic between prefixes X and Y," where X and Y belong to eyeball ISPs whose users are affected by the bad performance.

Fig. 3.1 illustrates the different units used in transparency: Witnesses observe and sample packets. The monitor defines traffic aggregates and maps each statement (hence each sampled packet) to an aggregate. As justified earlier (§3.2, No clean-Diesel measurements), at the time of forwarding, witnesses do not know which packets will be sampled, hence they cannot treat them preferentially.

3.5 Split-responsibility Means

Neighbor-dependent metrics. The monitor estimates each network's loss and delay w.r.t. individual traffic aggregates. Similar to packet delay, each network is held responsible for its internal delay plus half of the delay experienced on each of its inter-domain links.

Defining the loss metric is a bit more challenging: We first explore the case where users care (so conflicts happen) about the number of an aggregate's packets a network has lost. In this case, we simply define loss in a similar-to-delay way, i.e., each network is held responsible for its internal loss plus half of the loss experienced on each of its inter-domain links. We discuss whether this is fair and where an inter-domain link "starts or ends" in App. §B.1.1. Later in this section, we show how we can handle cases where it is not directly the number of lost packets but the loss rate over which conflicts happen. The challenge is that loss rates are not additive.

Per-aggregate lost packets. Consider a traffic aggregate A observed by a sequence of networks $\langle N_i,\ i \in \mathbb{N}\rangle$. The monitor estimates N_i's *split-responsibility lost packets w.r.t. aggregate* A as:

$$\widehat{LP}_i(A) \triangleq l + \frac{1}{2}\left(l^- + l^+\right), \tag{3.3}$$

where l is the number of A's sampled packets that were lost inside N_i, and l^- (resp. l^+) is the number of A's sampled packets that were lost on the inter-domain link before (resp. after) N_i.

Per-aggregate loss mean. The monitor estimates N_i's *split-responsibility loss mean w.r.t. aggregate* A as:

$$\widehat{LR}_i(A) \triangleq \frac{\widehat{LP}_i(A)}{m - \sum_{j=1}^{i-1} \widehat{LP}_j(A)}, \tag{3.4}$$

where m is the number of A's sampled packets that exited the source network N_1.

Per-aggregate delay mean. The monitor estimates N_i's *split-responsibility delay mean w.r.t. aggregate* A as:

$$\widehat{D}_i(A) \triangleq d + \frac{1}{2}\left(d^- + d^+\right), \tag{3.5}$$

where d is the average delay experienced by A's packets inside N_i, while d^- (resp. d^+) is the average delay experienced by A's packets on the inter-domain link before (resp. after) N_i. All averages are taken over the sampled packets observed by both witnesses of N_i.

Conflict as a consequence of lying. Given the threat-of-conflict over an aggregate's number of lost packets, Lemma 3.1 directly applies to per-aggregate lost packets. Lemma 3.1 also applies to delay means because means are additive, i.e., there is linearity of expectation. So: similar to packet loss/delay, a network has an incentive to be honest w.r.t. lost packets/delay mean of an aggregate. Otherwise, it necessarily shifts part of the blame for its internal lost packets/delay mean w.r.t. the aggregate to a neighbor and risks entering conflict. Further, the monitor can extract per-aggregate loss means from the accurately-reported lost packets and the number of

sampled packets that exited the source network (the source network has an incentive to accurately report this number because it is in its best interest to determine loss along the network path).

When conflicts won't happen over an aggregate's number of lost packets. Even when users do not directly care about the number of lost packets w.r.t. an aggregate but about the aggregate's loss mean, still we can redefine loss means such that a network has an incentive to honestly report them.

Per-aggregate pass-through mean. The monitor estimates N_i's *split-responsibility pass-through mean w.r.t. aggregate* A as:

$$\widehat{S}_i(A) \triangleq s\sqrt{s^- s^+}, \tag{3.6}$$

where s is A's pass-through mean inside N_i, i.e., the percentage of A's sampled packets that exited N_i with respect to A's sampled packets that entered N_i, and s^- (resp. s^+) is the pass-through mean of A's sampled packets on the inter-domain link before (resp. after) N_i.

Per-aggregate loss mean. The monitor estimates N_i's *split-responsibility loss mean w.r.t. aggregate* A as:

$$\widehat{LR}_i(A) \triangleq 1 - \widehat{S}_i(A). \tag{3.7}$$

Conflict as a consequence of lying. A network can lie about its pass-through mean w.r.t. an aggregate A only in one way: it can falsely blame some of its internal pass-through mean on one of its inter-domain links. This leads to the following:

Lemma 3.2 *Consider an aggregate A observed by a sequence of networks $\langle N_i,\ i \in \mathbb{N}\rangle$. If the monitor computes each network's pass-through mean w.r.t. A, and N_i's pass-through mean is increased compared to ground truth by a multiplicative factor σ_i, then $\prod_i \sigma_i = 1$.*

The gist of the proof (in §3.6.2) is same as for Lemma 3.1 (one subtle difference is that pass-through means are multiplicative instead of additive). As a result, a network cannot hide its loss mean, i.e., increase its perceived pass-through mean, w.r.t. an aggregate; it can only influence whether the loss appears to have happened before or after it.

For example, suppose N_i falsely claims a percentage σ^2 of A's pass-through mean on the inter-domain link after N_i as part of A's pass-through mean inside N_i. If N_{i+1} disputes, then there's conflict, and the monitor does not compute the performance of any network w.r.t. A. If N_{i+1} does nothing, then N_i's pass-through mean is exaggerated by a multiplicative factor σ, while N_{i+1}'s pass-through mean is exaggerated by a multiplicative factor $\frac{1}{\sigma}$ (penalized by $\frac{1}{\sigma}$).

Confidence intervals. The monitor can compute a confidence interval for the loss-mean estimate using one of the existing, standard loss models (App. §B.1.3). Computing a confidence interval for the delay-mean estimate, however, is significantly harder: there does exist a way (App. §B.1.3), but it requires an accurate estimate of N_i's jitter w.r.t. A.

3.6 Proofs

3.6.1 Lemma 3.1

W.l.o.g., we prove the lemma for packet delay and the network path $\langle N_{i-1}, N_i, N_{i+1}\rangle$, but the same principle also holds for larger network paths and packet loss.

Each network only controls the times of packet p at its entry and exit point. Hence, to exaggerate its delay w.r.t. p, each network must tamper with the statements it emits, which results in a certain amount of delay being pushed to its inter-domain links.

Let $\delta_i^- \in \mathbb{R}^+$ (resp. $\delta_i^+ \in \mathbb{R}^+$) denote the difference in p's delay of the previous (resp. subsequent) inter-domain link of N_i that is caused by N_i's dishonesty. Note that if $\delta_i^- = \delta_i^+ = 0$, N_i's statements are correct.

Based on Eq. (3.2), we now compute the differences between the delays d_i' that the monitor computes in the presence of dishonesty and d_i, which is what it would have computed if the networks were honest.

$$\delta_i \triangleq d_i' - d_i = -\frac{1}{2}(\delta_i^+ + \delta_i^- - \delta_{i-1}^+ - \delta_{i+1}^-) \tag{3.8}$$

$$\delta_{i+1} \triangleq d_{i+1}' - d_{i+1} = -\frac{1}{2}(\delta_{i+1}^- - \delta_i^+) \tag{3.9}$$

$$\delta_{i-1} \triangleq d_{i-1}' - d_{i-1} = -\frac{1}{2}(\delta_{i-1}^+ - \delta_i^-) \tag{3.10}$$

By summing up (3.8)–(3.10), we obtain the result.

3.6.2 Lemma 3.2

W.l.o.g., we prove the lemma for the network path $\langle N_{i-1}, N_i, N_{i+1}\rangle$, but the same principle also holds for larger network paths.

Each network only controls aggregate A's sampled packets at its entry and exit point. Hence, to exaggerate its pass-through mean w.r.t. A, each network must tamper with the statements it emits, which results in a certain percentage of the pass-through mean on its inter-domain links being pushed inside the network.

Let $\frac{1}{\sigma_i^-} < 1$ (resp. $\frac{1}{\sigma_i^+} < 1$) denote the multiplicative decrease in A's pass-through mean on the previous (resp. subsequent) inter-domain link of N_i that is caused by N_i's dishonesty. Note that if $\frac{1}{\sigma_i^-} = \frac{1}{\sigma_i^+} = 1$, N_i's statements are correct.

Based on Eq. (3.6), we now compute the ratios between the pass-through means s_i' that the monitor computes in the presence of dishonesty and s_i, which is what it would have computed if

the networks were honest.

$$\sigma_i \triangleq \frac{s'_i}{s_i} = \sqrt{\frac{\sigma_i^+ \sigma_i^-}{\sigma_{i-1}^+ \sigma_{i+1}^-}} \tag{3.11}$$

$$\sigma_{i+1} \triangleq \frac{s'_{i+1}}{s_{i+1}} = \sqrt{\frac{\sigma_{i+1}^-}{\sigma_i^+}} \tag{3.12}$$

$$\sigma_{i-1} \triangleq \frac{s'_{i-1}}{s_{i-1}} = \sqrt{\frac{\sigma_{i-1}^+}{\sigma_i^-}} \tag{3.13}$$

By multiplying (3.11)–(3.13), we obtain the result.

3.7 Summary

The proposed transparency protocols enable users to accurately assess the performance of individual networks but rely on conflicts that happen over individual packets. User interests, however, may concern larger traffic aggregates or different metrics, rendering existing solutions inaccurate. Ideally, a transparency protocol would adapt to the traffic units and metrics that users care about because it is only the dishonesty about those that leads to conflicts between networks.

We adapt incentives for honesty to the traffic units over which conflicts happen (e.g., traffic from certain video providers, or certain gaming platforms). We propose definitions of loss and delay metrics that leverage conflicts and provably incentivize honesty.

4 Policy-based Grouping of Traffic for Verifiable Jitter

Despite the importance of packet delay variance (jitter) for many applications, existing transparency proposals cannot track jitter issues down to specific networks: State-of-the-art transparency protocols have to assume threat-of-conflict incentives regarding individual packets and use sampling to reduce bandwidth overhead: networks report only on a small sample (e.g., 5%) of the packets they observe. These approaches are efficient for networks but at the cost of accuracy: they cannot reason about delay variance, because variance is not an additive metric, and one cannot build threat-of-conflict incentives for reporting it honestly; and they cannot reason about delay mean, because when delay mean is estimated from samples, the confidence interval is a function of variance.

In this chapter, we achieve accurate and efficient jitter estimation by combining threat-of-conflict incentives regarding user-defined aggregates of packets with mathematical tools. We identify an interplay between per-aggregate delay means and jitter: if a network exposes a set of traffic aggregates to the same network conditions, its delay means w.r.t. these aggregates and its overall delay variance cannot vary arbitrarily—they meet certain simple, well-defined mathematical constraints. From these constraints (and the honestly reported per-aggregate delay means), a monitor estimates each network's jitter. Evaluation through real traffic traces and simulation shows that Aether, our performance-transparency protocol, improves the monitor's delay-deviation estimates (jitter's square root), hence also the confidence intervals of its delay-mean estimates, by up to 3x in the presence of dishonest networks; and that it maintains its accuracy across a diverse set of network conditions—severe congestion, traffic shaping, and load-balancing—that make delay estimation hard.

4.1 Aether Overview

Aether, our performance-transparency protocol, makes it possible to accurately estimate networks' loss mean, delay mean, and jitter for user-defined aggregates of packets. This section provides an overview of the steps that the monitor and the networks' witness managers perform in Aether.

The threat model is the same as in §3.1.2.

(1) **Setup**: The monitor defines a set of traffic aggregates that traversed the networks in the target epoch. Moreover, the monitor collects the statements published by each network in the target epoch (from the network's witness manager) and maps each statement to an aggregate.

(2) **Split-responsibility means**: The monitor estimates each network's loss and delay mean w.r.t. the traffic aggregates defined in Step 1, such that each network is held responsible for its internal loss and delay and half the loss/delay of each inter-domain link (§3.5). For each metric it computes for a network, the monitor makes available to the network's witness manager the metric itself and all the statements used to compute it.

(3) **Check for conflict**: Each witness manager checks whether it agrees with the metrics computed by the monitor; if not, it declares to the monitor which statements it disputes. If network N_i's witness manager disputes a statement emitted by neighbor N_{i+1} w.r.t. a traffic aggregate A, the monitor publicly declares N_i and N_{i+1} to be "in conflict" and does not compute any network's performance w.r.t. A until the conflict is resolved. This happens when N_i and N_{i+1}'s witness managers both declare so to the monitor, potentially after updating their past statements.

(4) **Jitter**: For each network N_i that is not in conflict, the monitor estimates N_i's jitter (delay variance) w.r.t. the traffic aggregates defined in Step 1, or it declares that it does not have enough data to do so (§4.2).

4.2 Jitter

CLT as a potential solution. In Step 4, the monitor estimates each network's jitter. The monitor does not trust the networks' individual statements, it trusts only the split-responsibility means computed in Step 2 (§4.1). Hence, it must base its estimator solely on the latter.

Suppose network N_i exposes all packets that traverse witness pair j to a delay distribution $\mathscr{D}$ that has variance σ^2. According to the central limit theorem (CLT), if we take a set of n random samples from $\mathscr{D}$ that satisfy one of certain conditions (e.g., independence), the mean of this set is normally distributed around $\mathscr{D}$'s mean with variance $\frac{\sigma^2}{n}$. I.e., if one knows the variance of this delay mean, they can estimate σ^2.

Now consider a set of n packets from an aggregate A that traversed N_i at witness pair j; if we think of N_i's delays w.r.t. these packets as n random samples from $\mathscr{D}$, then the CLT opens the possibility for estimating jitter as a function of per-aggregate delay means. More specifically: If N_i's delays w.r.t n packets from A satisfied a CLT condition, then their mean should be normally distributed around $\mathscr{D}$'s delay mean with variance $\frac{\sigma^2}{n}$. Differently said: If a set $\mathscr{A}_j$ of n-sized aggregates traversed N_i at witness pair j, then N_i's delay means w.r.t. these aggregates (i.e., $\{\, \widehat{D}_A \mid A \in \mathscr{A}_j \,\}$) should be normally distributed with variance $\frac{\sigma^2}{n}$. Hence, the monitor could estimate σ^2 from $var\{\, \widehat{D}_A \mid A \in \mathscr{A}_j \,\}$.

But, can we apply any version of the CLT in our context? I.e., is there a subset of A's packets whose delays satisfy a CLT condition? And if so, does the implied normal distribution for the delay mean have a variance equal to $\frac{\sigma^2}{n}$?

PASTA as the missing link The answer to both questions is yes, if instead of a "static" delay distribution, we consider the continuous-time stochastic process that captures N_i's delay w.r.t. the traffic that traverses witness pair j.

▷ Let $\mathscr{D}$ be the converging sequence of N_i's packet delays as epoch length t tends to infinity, and σ^2 be $\mathscr{D}$'s variance (i.e., N_i's jitter).

▷ Consider an aggregate A traversing N_i at witness pair j, and a subset $A^* \subseteq A$, whose packets arrive at N_i as a Poisson process of rate λ. $\widehat{D}_{A^*}$ denotes N_i's delay mean w.r.t. A^*.

▷ According to the generalized PASTA theorem [21, Theorem 3] [68, 79]: as $t \to \infty$, $\widehat{D}_{A^*}$ converges to N_i's time-average delay μ, which can be perceived as the expected delay of a random packet (that traverses witness pair j) at N_i.

▷ According to the CLT for customer and time averages when PASTA holds [39, Proposition 5]: $\widehat{D}_{A^*}$ is normally distributed around μ with variance $(\lambda t)^{-1} \cdot \sigma^2$.

▷ By taking $\lambda = n/t$ (i.e., n samples per epoch), we coin:

Lemma 4.1 (CLT Condition) *Given a set of mild assumptions,*

$$\widehat{D}_{A^*} \sim \mathscr{N}\left(\mu, \frac{\sigma^2}{n}\right),$$

where μ is the time-average delay of a random packet arriving at N_i within epoch length t.

The proof and mild assumptions are listed in App. §A. The monitor relies on this lemma to estimate N_i's jitter σ^2 and to compute a confidence interval for its delay-mean estimates.

Jitter estimation. For each network N_i, the monitor groups together all the traffic aggregates that traversed N_i at the same witness pair. For each witness pair j, the monitor does the following:

(a) It sub-samples the corresponding aggregates and creates a set of *normalized aggregates* $\mathscr{A}_j$ that have the same number of packets n and (roughly) follow Poisson arrivals.

(b) It checks whether $\mathscr{A}_j$ is sufficient (it has enough aggregates and samples/aggregate) for accurately testing whether N_i exposed all the aggregates in $\mathscr{A}_j$ to the same delay process. If not, the monitor declares that it cannot estimate N_i's jitter w.r.t. the aggregates in $\mathscr{A}_j$, because it does not have enough data. Otherwise:

(c) It checks whether N_i exposed all the aggregates in $\mathscr{A}_j$ to the same delay process, using a standard hypothesis test for the CLT (e.g., a normality test). If not, the monitor again declares

that it cannot estimate N_i's jitter w.r.t. the aggregates in $\mathscr{A}_j$, because there is evidence of multiple delay processes. Otherwise:

(d) It estimates N_i's jitter w.r.t. each aggregate A' in $\mathscr{A}_j$ by inverting Lemma 4.1:

$$\hat{\sigma}^2_A = n \cdot \mathit{var}\{\, \widehat{D}_{A'} \mid A' \in \mathscr{A}_j \,\}. \tag{4.1}$$

No independence, Poisson assumptions. Independence is the most commonly used CLT condition in the static case, but we do not need it in our dynamic context. Moreover, we do not assume that the packets that N_i samples from each aggregate follow Poisson arrivals at N_i. Rather, the monitor explicitly picks a subset of the sampled packets, such that their arrival times at N_i form a Poisson process.

When jitter cannot be estimated. Even in the problematic scenarios where jitter cannot be estimated, network transparency is improved. The former scenario (not enough data) reveals that the problematic witness pair does not carry traffic of interest to network users that is of enough diversity (number of aggregates) and/or volume (aggregate size). The latter scenario (multiple delay processes) reveals that—with high probability—there was traffic differentiation (e.g., policing or shaping) between the problematic witness pair. The latter also opens the door to accurate assessment of a network's policy—a topic we discuss in §4.3.

4.3 Looking Ahead: Verifiable Network Policies

In this section, we outline how performance transparency could enable meaningful, verifiable network policies.

4.3.1 Policy Declaration

In our vision, each network declares a policy consisting of traffic classes (capturing the network's level of neutrality) and service-level agreements (capturing the network's performance w.r.t. each traffic class).

A **traffic class**, declared by network N_i, is a set of packets that traverse N_i at the same witness pair (same direction), and are exposed, within N_i, to the same network conditions, i.e., the same loss and delay processes. A network N_i declares its classes such that a packet belongs to one class and can be mapped to that class based on its headers. A typical policy could be to declare two classes per witness pair: a "latency-sensitive traffic" class and an "all other traffic" class (where "latency-sensitive" could be the traffic originating at a given set of content providers and gaming platforms). E.g., if N_i is a French eyeball ISP, it may declare a traffic class as "latency-sensitive traffic originating at (resp. addressed to) home networks in Ile-de-France and exiting (resp. entering) at the Equinix PoP in Paris."

A **service level agreement** (SLA) maps a traffic class to a promised loss mean, delay mean, and/or jitter over a given time period (e.g., seconds or minutes).

Traffic-class and SLA declaration does not incur any overhead nor force networks to change their policy. A network that does not apply any traffic differentiation nor commits to any SLA simply declares a single traffic class ("all traffic") for all its witness pairs and does not associate any SLA with that class.

4.3.2 Policy Verification

The monitor could verify such network policies by re-using and extending the techniques presented earlier: It would define traffic aggregates and compute each network's per-aggregate loss and delay mean exactly as in §3.5. For each network N_i, instead of grouping the traffic aggregates that traversed N_i per witness pair, it would group them per traffic class. Then, for each traffic class, the monitor would: (a) check whether N_i exposed all traffic aggregates that belong to the class to the same network conditions and, if yes, (b) estimate N_i's jitter—using the PASTA/CLT-based techniques described in §4.2. Finally, for each traffic class, the monitor would estimate N_i's overall loss and delay mean (by averaging the per-aggregate means) and check whether they conform to the specified SLAs.

When network policies cannot be verified. The monitor would not always be able to verify all network policies, in part because of the same reasons it cannot always estimate jitter: the monitor may not have enough data, or the data does not satisfy the proper conditions (§4.2). Alternatively, a network may abuse policy declaration to make it impossible for the monitor to verify its policy. For example, network N_i may declare more traffic classes that it actually has such that each class contains a small number of traffic aggregates (one aggregate per class, in the extreme). Such a strategy, however, would be transparent and to the detriment of N_i: declaring more than a handful of traffic classes per witness pair indicates an unreasonable level of traffic differentiation; even if there are no neutrality regulations, neutrality violation is never popular with network users, and we doubt that a network would publicly declare lots of little traffic classes in order to evade the assessment of its performance.

Still, even when network policies cannot be verified, besides per-aggregate loss and delay means, is there anything that the monitor can accurately infer about a network's performance?

We identify a connection between aggregate delay means, the overall delay variance of a network, and traffic differentiation. As a result, a network that honestly reports its aggregate delay means cannot hide its overall delay variance, it can only explain it as non-neutrality.

Class disparity is variance. Consider all aggregates that traverse network N_i; suppose N_i's delay means w.r.t. these aggregates vary significantly relative to each other. This variance is due to: (1) variable network conditions that apply to all traffic alike, e.g., congestion, routing oscillations, transient infrastructure failures; and/or (2) traffic differentiation, e.g., aggressively

shaping certain aggregates, or routing them through slower/more loaded paths.

The law of total variance. Once we think of variable network conditions and traffic differentiation as different factors that contribute to delay variance, we can connect them through the law of total variance: Consider an aggregate that traverses network N_i; let X_A be a r.v. with finite variance, representing N_i's split-responsibility delay mean w.r.t. the aggregate; let X_C be a r.v. representing the aggregate's traffic class; according to the law of total variance, we can write:

$$\mathrm{Var}(X_A) = \mathrm{E}(\mathrm{Var}(X_A|X_C)) + \mathrm{Var}(\mathrm{E}(X_A|X_C)),$$

where E and Var represent, respectively, expectation and variance. $\mathrm{Var}(X_A)$ is N_i's overall delay variance, i.e., N_i's delay variance w.r.t. all aggregates. The first right-hand side term is what statisticians call "unexplained variance": $\mathrm{Var}(X_A|X_C)$ is N_i's delay variance w.r.t. aggregates of class X_C; if the value of X_C is not fixed, then $\mathrm{Var}(X_A|X_C)$ is itself a r.v., whose expectation summarizes delay variance within classes (which is due to variable network conditions). The second right-hand side term is what statisticians call "explained variance": $\mathrm{E}(X_A|X_C)$ is N_i's expected delay w.r.t. class X_C; if the value of X_C is not fixed, then $\mathrm{E}(X_A|X_C)$ is itself a r.v., whose variance summarizes delay variance across classes (which is due to traffic differentiation).

Jitter. The monitor computes network N_i's *jitter* as an empirical version of the unexplained term in the law of total variance:

$$\text{Jitter} = \mathrm{E}(\mathrm{Var}(X_A|X_C)),$$

where the variance is over all aggregates in class X_C and the expectation is over all classes in the network's policy. This metric summarizes delay variance within traffic classes.

Class disparity. The monitor computes network N_i's *class disparity* as an empirical version of the explained term in the law of total variance:

$$\text{Class disparity} = \mathrm{Var}(\mathrm{E}(X_A|X_C)),$$

where the expectation is over all aggregates in class X_C and the variance is over all classes in the network's policy. This metric summarizes delay variance across traffic classes.

Incentives. A network cannot hide its overall delay variance; it can only choose whether the monitor perceives it as jitter or class disparity. This leads to the following:

Lemma 4.2 *If network N_i lies about its jitter by $-\delta$, it will also lie about its class disparity by $+\delta$.*

The lemma is a direct consequence of the law of total variance: N_i may advertize as many classes as it wants (independently of whether it exposes all traffic from the same class to the same network conditions). However, according to the law of total variance, grouping aggregates in different numbers of classes does not change the sum of the unexplained (jitter) and explained

(class disparity) variances. Hence, by lying about the former by $-\delta$, N_i is lying about the latter by δ.

4.4 Experimental Evaluation

After describing our experimental setup (§4.4.1), we summarize Aether's overhead (§4.4.2), evaluate its accuracy (§4.4.3 and §4.4.4), and show evidence of its potential for network-policy verification (§4.4.5).

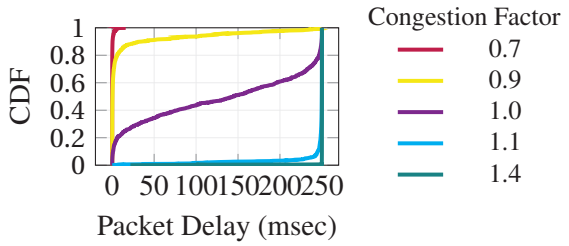
4.4.1 Experimental Setup

Rationale. The generalized PASTA and CLT theorems reason about the convergence of a stochastic process over *infinite time*; in practice, they are applied to finite, "sufficiently long" time intervals—how long is "sufficiently long" depends on the context. In our context, the stochastic process in question is the delay of a given network w.r.t. packets that traverse a given witness pair; and the convergence concerns a subset of an aggregate's samples that arrive at the given network at Poisson-obeying time instants. We must answer: does this process reach this convergence *within an epoch*, which may last from several seconds to several minutes? For this, we must reproduce two elements from a real network: its delays w.r.t. the packets that traverse a witness pair; and the arrival times of packets from individual traffic aggregates. We obtain the former from simulation, and the latter from CAIDA traffic traces [3].

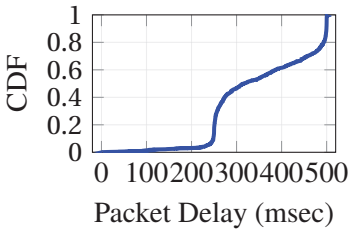
Simulated paths. In each experiment, we simulate one internal network path (i.e., one witness pair) of one network N_i. We simulate four path types: "single-queue" consists of a single queue/link, "cross-traffic" consists of six queues/links (akin to Google's B4 paths), "traffic-shaping" consists of a queue/link equipped with a traffic shaper, and "load-balancing" consists of two parallel queues/links. We use the first one to simulate congestion due to a single bottleneck; the second one for congestion due to cross-traffic; the third one for non-neutral conditions; and the fourth one to simulate parallel, independent bottlenecks during internal load-balancing. The buffer size of each link is configured according to the round-trip time (RTT) rule of thumb, using RTT=250msec, unless otherwise stated.

Input traffic. In each experiment, we feed a traffic trace to a simulated path. We use 21 one-hour CAIDA traces [3], with incoming trace rates ranging from 1.9 to 4.5Gbps. Specifically, we use from these traces: (1) the packet timestamps to feed the packets into a simulated path, (2) the packet sizes when buffering, transmitting, or shaping packets, and (3) the packet source/destination IP addresses when defining traffic aggregates. In "cross-traffic," we feed half of the traffic aggregates to the simulated path, while the other half cross the simulated path only at the third link. In "traffic-shaping," half of the traffic aggregates ("low" priority) go through the traffic shaper, whereas the other half ("high" priority) do not; "low" is shaped such that it achieves 90% or 95% of the rate of "high". In "load-balancing," packets randomly take one of the two parallel

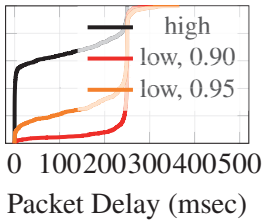
paths.



(a) Single-queue path under different congestion conditions



(b) Cross-traffic path



(c) Traffic-shaping path under different shaping ratios

Figure 4.1 – CDF of packet delay.

Witness configuration. The witnesses sample 5% of the packets they observe, which results in 0.20% bandwidth overhead, unless otherwise stated. We picked it in order to keep the bandwidth overhead "well below 1%" according to the informal recommendation of a network operator.

Monitor configuration. The monitor divides the 21 hours of simulated time into non-overlapping equilength epochs (length varies from 16sec to ~17min across configurations). For each epoch, the monitor defines a traffic aggregate for each /24 source-destination prefix pair; this serves as a proxy for "all packets between the same edge networks" and stresses Aether's accuracy: /24 aggregates have fewer packets compared to coarser ones, hence it's harder to extract reliable statistics from them. At the end of each epoch, the monitor estimates network N_i's per-aggregate loss mean, delay mean, and jitter. To test whether a set of aggregates were exposed to the same delay process, the monitor uses the Shapiro-Wilk (SW) hypothesis test to assess normality. We set the confidence level of the test to 99%.

4.4.2 Is it Lightweight?

We now estimate the overhead incurred by the participating networks and the resources required to implement a monitor.

Networks. Each participating network computes statements (at the witnesses) and exports them to the witness manager. The cost of computing the statements is the cost of performing consistent sampling with delayed disclosure: in terms of data-plane memory, it is a small (up to 10%) increase in data-plane memory; in terms of computation, it consists of two lightweight hash

Packet Sampling Rate (%)	Bandwidth Overhead (%)
1.00	0.04
5.00	0.20
10.00	0.40

Table 4.1 – Bandwidth overhead incurred by a network.

functions and a lookup in a small TCAM per observed packet [66]. The cost of exporting the statements depends on the witnesses' sampling rate, the average packet size they observe, and the size of each statement. Table 4.1 shows the resulting bandwidth overhead for three example sampling rates, average packet size of 891 bytes (as in the CAIDA traces), and a statement size of 18 bytes (4 bytes for packet identifiers, as in [66], 4 bytes for timestamps—msec granularity and one-month wraparound—6 bytes for /24 source and destination IP prefixes, and 4 bytes for port numbers, ignoring any applicable compression).

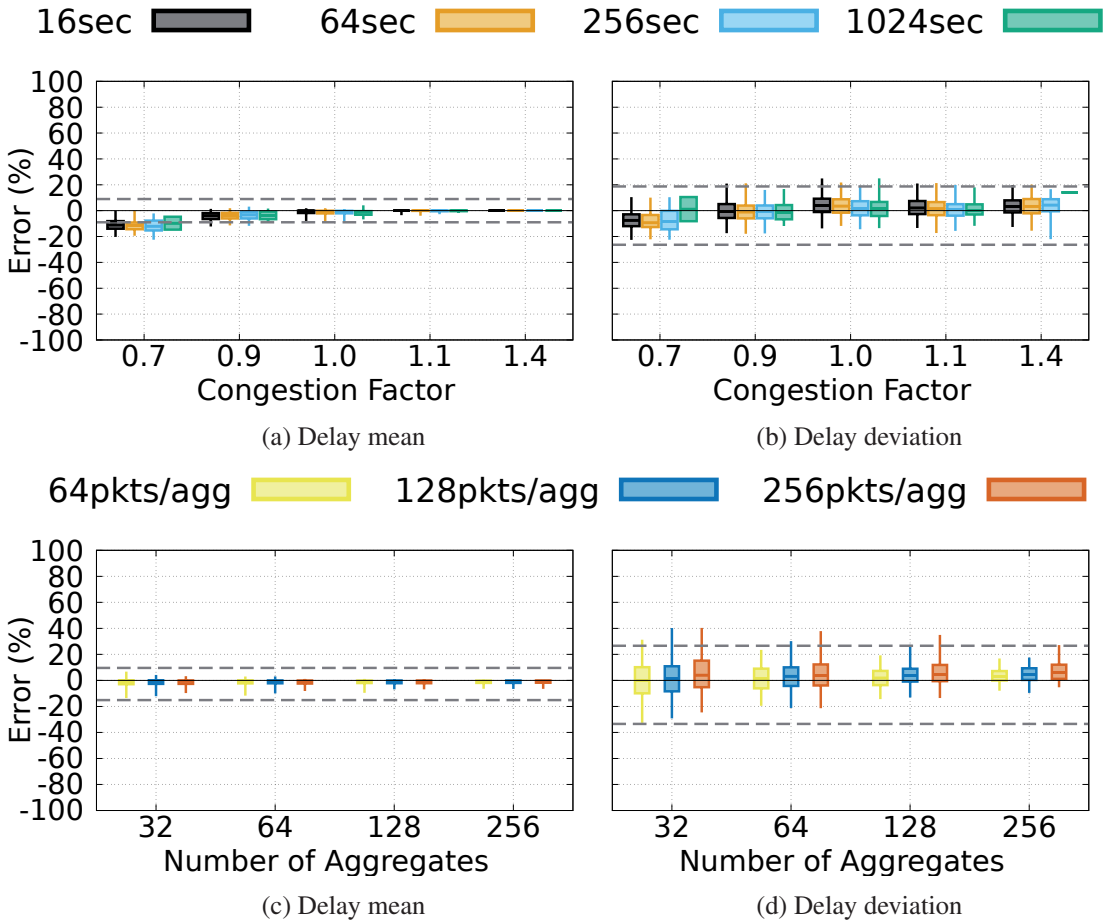
Monitor. We assume a monitor that keeps a list of all eyeball and transit ISPs, divides time in epochs, and randomly picks, during each epoch, one target ISP to focus on. In each epoch, the monitor defines traffic aggregates that traverse the target ISP, identifies all the witnesses (of the target ISP and other networks) traversed by these aggregates, collects their statements, and performs the steps in §4.1. We implemented the monitor in C++ and ran it on a dual-socket Xeon CPU E5-2680 (12 cores per socket @ 2.50GHz). We found that 50 cores are enough to handle the load of such a monitor in real time (i.e., complete all the steps before the next epoch begins) (see App. §B.2.1 for the details).

4.4.3 Is it Accurate?

We first assess the accuracy of the monitor's estimates when networks are honest.

Error types. The monitor incurs two types of estimation error: *sampling error* (incurred by all estimates), which is due to the fact that estimation is based on a sample from each aggregate; and *convergence error* (incurred by the jitter estimates), which is due to the fact that estimation assumes PASTA/CLT convergence, which may not occur during the target epoch. We focus our analysis on the jitter estimates, given that the other two types incur only sampling error, which is well understood. We also show delay-mean estimates—for completeness, and such that the jitter numbers make sense.

Simulated congestion. First, we run a set of experiments where we simulate a "single-queue" path, while varying the "congestion factor," epoch length, number of traffic aggregates, and aggregate size (number of samples). The "congestion factor" is the average arrival rate of the input traffic divided by the transmission rate of the bottleneck link. Fig. 4.1 shows the empirical packet-delay CDFs for the various congestion factors we examine; the wide range of these distributions (that also include extremes, e.g., congestion factors 0.7 and 1.4) allows us to



(a) Delay mean

(b) Delay deviation

(c) Delay mean

(d) Delay deviation

Figure 4.2 – Estimation accuracy w.r.t. congestion level & epoch length (a, b), and aggregates' cardinal number & size (c, d).

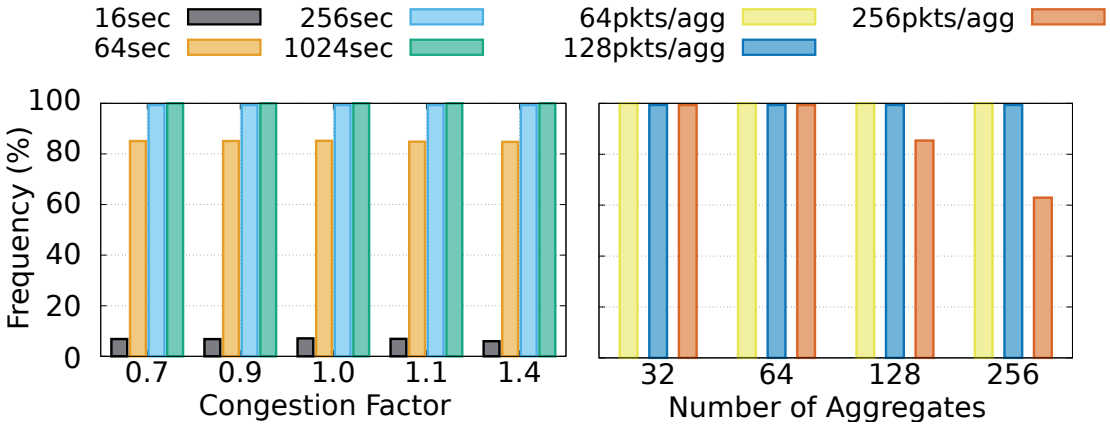
thoroughly test our estimation method, even at its operating limits.

Default aggregate number and size. The monitor performs two tasks affected by aggregate number and size: a normality test (Step 4c in §4.2) and jitter estimation (Step 4d). For each task, the monitor is free to use as many of the defined aggregates, and as many of the sampled packets per aggregate as it wants. It turns out that the best aggregate number and size is different for each task: for jitter estimation, more and bigger aggregates is better (though the improvement is less marked beyond 128 aggregates and 64 sampled packets per aggregate); for the normality test, it is better to use relatively fewer aggregates that are as large as possible (due to an artifact of the SW hypothesis test). Hence: for jitter estimation, our default configuration is 128 aggregates and 64 sampled packets per aggregate; for the normality test, it is 32 aggregates and 256 packets per aggregate. We further discuss sensitivity to aggregate number and size in App. §B.2.2.

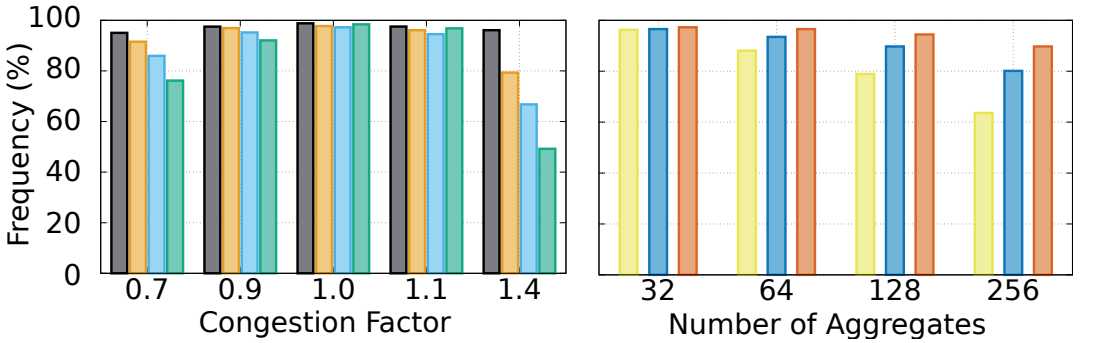
Accuracy metrics. For each simulated epoch, we compute two summary accuracy metrics: (1) *Delay-mean error*: we average the monitor's delay-mean estimates (across all the aggregates),

and we compute the relative error between this overall delay-mean estimate and the ground truth (the true delay mean experienced by all traffic across the simulated path). (2) *Delay-deviation error*: we compute the relative error between the monitor's delay-deviation estimate and the ground truth (the true delay deviation experienced by all traffic across the simulated path). We plot delay-deviation (jitter's square root) as opposed to jitter error, because this is easier to relate to delay-mean error.

Presentation. Fig. 4.2 summarizes the results in 4 boxplots. Each box corresponds to a different experiment, and it shows the 1st, 25th, 50th, 75th, and 99th percentile of the monitor's delay-mean or delay-deviation error. In each boxplot, the horizontal dotted grey lines show the 1st and 99th percentile of the monitor's error (across all experiments represented in the boxplot), in the hypothetical scenario where packets are grouped into traffic aggregates truly randomly (as opposed to based on IP prefixes). These lines show how accurate our estimation would be in an ideal scenario where the CLT applied trivially, and how far our actual accuracy is from that ideal.



(a) Are there enough data?



(b) Can we apply the CLT?

Figure 4.3 – How often can the monitor estimate jitter.

Estimation error. Fig. 4.2 shows that Aether is accurate across diverse scenarios (assuming reasonable aggregate numbers and sizes). Fig. 4.2a shows the monitor's delay-mean error, while Fig. 4.2b its delay-deviation error, for different congestion factors and epoch lengths; in these

experiments, the monitor relies on 128 traffic aggregates and 64 sampled packets per aggregate. We see that the 75th percentile of the error stays below 10%, while the 99th percentile stays below 25%. The worst-case (near 25%) appears for very low congestion (Fig. 4.2b, congestion factor 0.7). In this scenario, the underlying delay distribution is so skewed (<10% of packets experience >1msec delay) that PASTA/CLT convergence does not always occur within an epoch. We discuss error sensitivity to aggregate number and size in App. §B.2.2.

Frequency of jitter estimation. Fig. 4.3 shows that the monitor succeeds in computing a jitter estimate across diverse scenarios. Fig. 4.3a shows how often the monitor finds the target aggregate number and size; Fig. 4.3b shows how often the CLT-based normality test succeeds (i.e., there is no evidence of multiple delay processes). The sub-figures on the left show these frequencies for different congestion factors and epoch lengths; in these experiments, the monitor relies on 32 traffic aggregates and 256 sampled packets per aggregate. We see that the monitor succeeds at least 78% of the time—as long as an epoch lasts at least 64sec, and congestion factor is up to 1.1. For shorter epochs, the observed traffic volume and diversity are not enough, i.e., the simulated path does not observe 32 distinct aggregates that are large enough to yield 256 sampled packets each (Fig. 4.3a, epoch length 16sec). For congestion factor 1.4, the skew-ness of the underlying delay distribution (Fig. 4.3b) is such that, if one observes the packet delays only within one epoch, it may look as if there are multiple delay distributions (i.e., PASTA/CLT convergence may take longer than an epoch). Still, the monitor succeeds in computing a jitter estimate at least 44% of the time. We discuss success-rate sensitivity to aggregate number and size in App. §B.2.2.

Beyond single-bottleneck paths. We also run a set of experiments where we simulate "cross-traffic," "traffic-shaping," and "load-balancing" paths (the latter with varying congestion factors). Fig. 4.4 shows that Aether maintains its accuracy across path types, despite the fact that some of them yield twice the max delay as single-queue paths (Figs. 4.1b and 4.1c). We plot the 1st (dashed lines) and 99th (solid lines) percentiles of the monitor's delay-deviation error for different path types and epoch lengths. In these experiments, the monitor relies on 128 traffic aggregates and 64 sampled packets per aggregate; in the "single-queue" and "load-balancing" experiments represented in this figure, the congestion factor is 1. We see that the lines corresponding to different path types mostly overlap—error does not worsen because of path type. (Also: "single-queue" and "load-balancing" experiments of the same congestion factor yield similar estimation errors—not shown.)

Comparison to prior work. Prior work does not estimate split-responsibility means or jitter. However, the state of the art [20, 66] does estimate each network's internal per-aggregate loss and delay means based on sampled packets from each aggregate. Assuming honest networks, the accuracy of these "conventional" estimates should be the same as the accuracy of Aether's split-responsibility estimates (since all of them are standard sampling-based estimates).

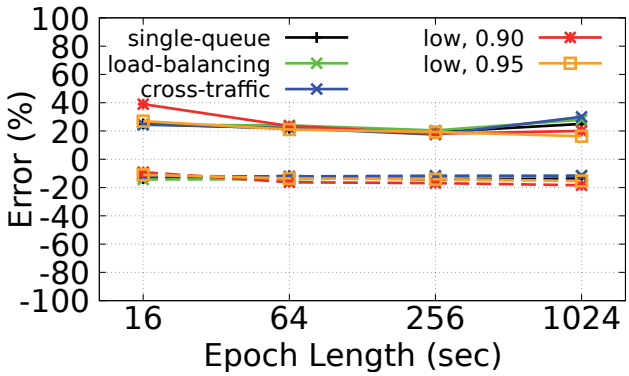


Figure 4.4 – Impact of path characteristics.

4.4.4 How much could one lie without it?

Aether enables the monitor to estimate jitter (hence also to reason about the accuracy of its delay-mean estimates) assuming networks may lie. We now quantify this benefit.

Lying strategy. We run a set of experiments where networks N_i and N_{i+1} lie in collaboration such that: *they exaggerate their jitter as much as possible without affecting their delay means*. More specifically: The two networks jointly run an optimization algorithm that alters N_i's exit and N_{i+1}'s entry timestamps, so as to minimize both networks's perceived jitter without affecting either of their delay means. We cast this as a convex optimization problem and solve it using the CVX framework [42, 43]. In summary, what the networks do is fake their internal delays to make them look more correlated than they truly are, which then makes it possible for both of them to claim lower jitter without affecting their delay means.

Comparison to prior work. We implement an alternative transparency protocol, where the monitor estimates network N_i's per-aggregate loss mean, delay mean, *and jitter* based on individual statements. This is essentially the state of the art [20, 66] but extended to perform standard, sampling-based jitter estimation.

Fig. 4.5 compares Aether (blue triangles) against this alternative (red crosses). The y-axis represents the monitor's delay-deviation error during different epochs. The x-axis represents the ratio between N_i's and N_{i+1}'s delay deviations; we vary this ratio, because different ratios yield different lies, and we want to show the entire range covered by the lying strategy. The shaded red area shows the potential of the lying strategy: in theory, as soon as the ratio of delay deviations increases beyond 0.5, networks can exaggerate their delay deviation by 100% (i.e., claim 0 jitter) without affecting their delay means or the estimates of their neighbors.

Fig. 4.5 shows that Aether reduces the monitor's delay-deviation error by up to 3.6x. The alternative protocol enables N_i to exaggerate its delay deviation by as much as 65%. With Aether, the monitor incurs only the convergence error we studied in §4.4.3, which is up to 18% in these experiments (up to 25% in the experiments we saw earlier).

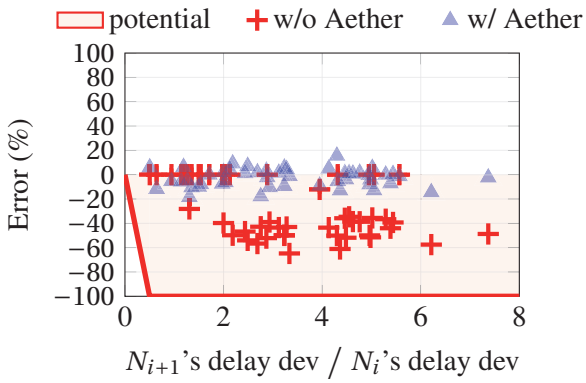


Figure 4.5 – Comparison of the monitor's accuracy in estimating the delay deviation of N_i with and without Aether.

4.4.5 Would it detect policy violation?

We claimed that Aether could be easily extended to verify network policies (§4.3); we now back this with a representative experiment. We use a "traffic shaping" path to simulate the scenario where network N_i declares one traffic class but actually differentiates between a high-priority and a low-priority class (e.g., N_i could be Cogent throttling Netflix traffic that transits its network). Would the monitor catch this policy violation? Fig. 4.6 shows that it would, at least 80% of the time and across a wide range of epoch sizes; and this, despite the fact that the shaping ratio is merely 90-95%! For more intense shaping, accuracy approaches 100%. This indicates the potential of using PASTA/CLT to detect even subtle traffic differentiation.

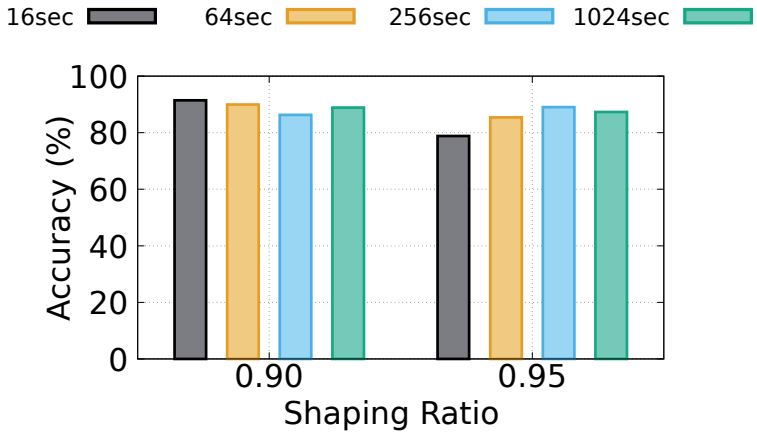


Figure 4.6 – Detecting false policy declarations.

4.5 Discussion

Deployment incentives. One avenue would be regulation: given that several countries (at least in Europe and America) already have regulations that require networks to be transparent about their traffic policies, it is not unreasonable to require that they participate in a transparency protocol.

Our preferred avenue, however, would be that some networks willingly participate because that enables them to showcase their great traffic policies, and that the rest of the networks follow because their customers come to expect it.

Partial deployment. Aether would "work" even if a single network (let's call it N_0) deployed it. We put "work" in quotes, because the monitor would be forced to accept all of N_0's statements (no neighbor would ever dispute them). That, however, could act as an incentive for N_0's neighbors to also deploy—otherwise N_0 would be free to blame any of its internal loss and delay on them. In short: under partial deployment, the monitor simply accepts all the statements that cannot be disputed (because there's no deployed witness at the other end of the line), and this enables participating networks to lie until their neighbors join the deployment.

More lightweight sampling. Aether incurs minimal bandwidth overhead (§4.4.2), and the monitor could be implemented on a few off-the-shelf servers. The main hurdle to deployment is the fact that participating networks must deploy a new sampling algorithm (consistent sampling with delayed disclosure) at each entry and exit point. Future work can potentially relax this requirement. Witnesses can continue performing traditional sampling (which, on its own, might lead to clean-Diesel measurements), and the monitor will be checking whether the sampled packets experience the same *end-to-end* loss and delay distributions as other packets (with help from edge networks); if yes, that means that the monitor can base its evaluation on the performance experienced by the sampled packets.

4.6 Related Work

Early transparency mechanisms. The earliest work is Packet Obituaries [18], where witnesses emit per-packet statements. In modern network devices, there is only modest bandwidth between the data plane and the local control plane. Increasing this bandwidth to support the continuous export of per-packet statements would require a significant shift in hardware design. In AudIt [19], witnesses emit per-TCP-flow statements. This is cheaper, but it still requires per-TCP-flow state (counters and timestamps); moreover, it makes networks vulnerable to denial of service, where an attacker sends single-packet flows causing witnesses to emit per-packet statements.

Sampling-based transparency mechanisms. The AudIt paper suggests using sampling to reduce overheads, and subsequent proposals explored this approach: in [81], intermediate nodes on the path send authenticated reports for a fraction of the packets they observe back to their sources, so that the latter can identify malfunctioning links; in [20] and [66], witnesses emit statements for a few sampled packets (same as in our proposal), and a monitor computes each network's average performance w.r.t. different traffic streams. These proposals, however, focus on loss and are silent on delay variance, which makes them vulnerable to networks that want to lie about delay. Our work bridges this gap by identifying a connection between per-aggregate delay means and delay variance that relies on the CLT and PASTA theorems.

Data-plane fault localization. We share goals with proposals that localize loss (or tampering) of specific packets on specific links. For example, ShortMAC [82] relies on intermediate nodes embedding localization information in the observed packets. This involves packet modification and might be a harder mechanism to deploy. However, one could combine such a mechanism with sampling to reduce overheads, in which case our results would be directly applicable. Also, [22] relies on intermediate nodes maintaining per-path state (secure sketches [40]), where a "path" could correspond, for instance, to a source-destination IP prefix pair. This approach is not suitable for our context, because the nodes would need to monitor all the paths all the time, which would be too expensive. Another interesting domain that we share common vision with is network provenance [85, 86], where a set of nodes maintain a distributed provenance graph that records important network events—potentially including packet arrivals and departures—and is used for diagnosis and forensics. Due to its high overhead, this approach seems most suitable for the control plane, and our sampling method could help the system scale.

Inter-domain routing fault localization. We share philosophy with proposals that localize faults in distributed systems in general [44], and inter-domain routing in particular [45]. In this work, each message (resp. routing message) to/from a node of a distributed system (resp. router) is logged together with timing information and cryptographic evidence of correct recording. This is akin to a transparency mechanism "logging" the transmission or reception of each packet. At the same time, observed packets are orders of magnitude more than routing messages; hence, state-of-the-art transparency mechanisms avoid expensive cryptographic operations upon packet transmission and focus on how to reduce overheads via sampling without losing reliability, which is not a concern in inter-domain routing.

4.7 Summary

When networks produce statements on their own performance (and may be dishonest), what can users accurately estimate from network statements? We show that it is possible to estimate not only loss and delay mean, but also delay variance—useful in itself, as well as necessary for computing confidence intervals for delay-mean estimates. With a <1% bandwidth overhead, we show that accurate estimation is feasible in the presence of dishonest networks, flexible user interests, and across challenging network conditions that normally make reasoning about delay hard.

We argue that our approach opens the door to meaningful traffic policies: the moment networks participate in a transparency protocol that makes it possible to accurately estimate their loss mean, delay mean, and jitter for individual traffic aggregates, it becomes straightforward for networks to declare meaningful traffic classes and SLAs, and for the relevant entities to verify that the latter are honored.

5 Adaptive Traffic Reports for Anonymous Communications

Can we improve Internet performance transparency without worsening user anonymity? For a long time, researchers have been proposing transparency protocols, where traffic reports produced at strategic network points help assess network behavior and verify service-level agreements or neutrality compliance. However, such reports necessarily reveal when certain traffic appeared at a certain network point, and this information could, in principle, be used to compromise low-latency anonymity networks like Tor.

In this chapter, we examine whether more transparency necessarily means less anonymity. We start from the information that a basic transparency solution would publish about a network and study how that would impact the anonymity of the network's users. Then we study how to change, in real time, the time granularity of traffic reports in order to preserve both user anonymity and report utility. We evaluate with real and synthetic data and show that our algorithm can offer a good anonymity/utility balance, even in adversarial scenarios where aggregates consist of very few flows.

5.1 Introduction

The Internet does not provide enough performance transparency: when traffic is lost, delayed, or damaged, there is no systematic way for the affected parties to determine where the problem occurred and which network is responsible. Lack of transparency has led to problems: Internet service providers (ISPs) sign service-level agreements (SLAs), committing to a certain packet delivery rate or latency [1, 5], that are impossible to verify; countries and regulatory bodies enact network neutrality regulations, requiring that ISPs treat all traffic the same independently from origin or application [8, 10, 24], that are impossible to enforce; content and network providers enter disputes that are impossible to investigate properly [6]. Note that improving transparency does not imply arguing for SLAs or regulations; but from the moment Internet users care enough for them to exist, there should be a way to verify and enforce them.

To improve Internet performance transparency, researchers have proposed "transparency proto-

cols," where networks report on their own performance [18–20, 22, 40, 66, 81, 82]. The common idea behind these proposals is that participating networks deploy special nodes at their boundaries, which collect samples or summaries of the observed traffic and report them to a monitor; based on these reports, one can accurately estimate each network's performance with respect to various traffic aggregates. The challenges typically addressed are how to produce useful reports at low cost, and how to prevent networks from exaggerating their perceived performance.

However, transparency protocols face another significant challenge, which, to the best of our knowledge, has not been addressed: they interfere with anonymous communications. Anonymity networks enable a user to hide the fact that she is communicating with a particular destination. Transparency protocols directly threaten this capability because they expose information about when given traffic is observed at given network points. For example, consider: a user that communicates with various destinations through Tor [14, 27]; a government that monitors the user's Internet connection and observes the flows she generates; and a basic transparency protocol, where participating networks periodically group flows into aggregates and report per-aggregate packet counts. If the government gains access to the information stored in the monitor, it becomes equivalent to a passive adversary that observes the user's flows, on the one hand, and all Tor aggregates, on the other, and tries to de-anonymize the user's flows; low-latency anonymity networks like Tor are vulnerable to such adversaries [26, 48, 60].

Is it possible to design a transparency protocol that produces useful reports without interfering with anonymous communications? On the one hand, there exists a fundamental trade-off between report utility and flow anonymity: the finer the time granularity of the reports, the better one can compute loss burstiness or delay jitter between reporting nodes (so, higher report utility), but also the better one can match flow and aggregate patterns (so, lower flow anonymity). On the other hand, even time granularity of minutes can be useful: in the current Internet, downloading movies or kernel distributions can take from minutes to hours; hence, reports that help determine whether ISPs honor their SLAs or honor neutrality at such time granularity are still useful. The question then is: how coarse do reports have to be in order to preserve flow anonymity? If a transparency protocol reports at a time granularity of minutes, is that enough? How about seconds?

We take a first step toward answering this question: We consider a basic transparency protocol, where nodes report per-aggregate packet counts, which can be used to compute packet loss between pairs of nodes (§5.2). We first study how such a protocol would affect flow anonymity in the context of a low-latency anonymity network (§5.3). Then we propose MorphIT, an algorithm that takes as input a node's report and produces as output a modified version that improves flow anonymity (§5.4). Our algorithm does not introduce noise in the typical sense, i.e., does not introduce error in the reported packet counts; instead, it merges packet counts so as to obfuscate the flow patterns that stand out the most within each aggregate. We evaluate this approach with synthetic and real traffic traces obtained from CAIDA (§5.5). We show that, even in highly adversarial scenarios (e.g., only 64 flows per aggregate), merging packet counts across sub-second time intervals is enough to prevent an adversary from tracing any flow to a unique aggregate, and it also significantly reduces the number of flows that can be traced to few candidate aggregates

(e.g., fewer than 5). Reporting packet counts of sub-second granularity means that we retain report utility, in the sense that we can use the reports to verify SLA or neutrality violations at such a fine granularity.

We close the chapter by discussing the limitations of our approach (§5.6), related work (§5.7), and our conclusions (§5.8).

5.2 Setup

After defining the basic terms we use in this chapter (§5.2.1), we state our threat model (§5.2.2) and problem (§5.2.3), and we summarize the most relevant anonymity metrics from related work (§5.2.4).

5.2.1 Definitions

We consider a transparency protocol that involves the same participants as in the previous chapters: networks, witnesses, and a monitor.

A **time tick** t is a time interval of the smallest duration for which a witness can produce statistics. A **time bin** T is a time interval of one or more time ticks. The **observation window** W, of size w, is the time interval for which our adversary (defined below) collects information.

A **flow** f is a sequence of packets exchanged between a unique source and destination. When we say that an adversary "knows a flow's pattern", we mean that she knows the packet inter-arrival times, but not necessarily the packet contents (because they could be encrypted) or the flow's destination (because the source could be using an anonymity network). $N_f(t)$ ($N_f(T)$) denotes the number of f's packets observed at a witness during time tick t (time bin T). λ_f denotes f's average packet count per time tick.

An **aggregate** A is a sequence of packets with a unique source and destination IP prefix that are observed at a particular witness. $N_A(t)$ ($N_A(T)$) denotes the number of A's packets observed during time tick t (time bin T). λ_A denotes A's average packet count per time tick.

A **traffic report** $R(A)$ refers to a specific aggregate A and is a set of tuples:

$$R(A) \triangleq \{\, \langle t, N_A(t) \rangle \,\}. \tag{5.1}$$

The reports produced by two different witnesses for the same aggregate A are used to compute the packet-loss rate between the two witnesses with respect to A.

There exist many flavors of transparency protocols [18–20, 22, 40, 66, 81, 82] each one reporting slightly different statistics. We consider the flavor that produces minimal information: a transparency protocol where witnesses produce per-aggregate packet counts. If this simple

transparency protocol poses a threat to flow anonymity, then any of the more sophisticated transparency protocols that have been proposed will also pose a threat.

Table 5.1 lists the symbols used in this chapter.

Symbol	Description
Traffic units	
f	A flow: a packet sequence exchanged between a unique source and destination
A	An aggregate: a packet sequence observed at a witness
$\mathscr{A}$	A set of aggregates
Time intervals	
t	A time tick
T	A time bin: a set of consecutive time ticks
$\mathscr{T}$	A set of consecutive, non-overlapping time bins
W	The adversary's observation window
$w = \lvert W \rvert$	The size of the adversary's observation window (in time ticks)
Traffic characteristics	
$N_f(t)$	Number of packets in flow f in time tick t
$N_f(T)$	Number of packets in flow f in time bin T
λ_f	Average packet rate of flow f
$N_A(t)$	Number of packets in aggregate A in time tick t
$N_A(T)$	Number of packets in aggregate A in time bin T
λ_A	Average packet rate of aggregate A
ϕ	Maximum number of active flows per aggregate
Algorithm parameters	
ρ	Max flow burst size per time tick (in packets)
τ	Max bin size (in time ticks)
ω	Active window size (in time ticks)
Differential Privacy	
ϵ	Privacy loss
δ	Probability of exceeding ϵ

Table 5.1 – List of symbols used in this chapter.

5.2.2 Threat Model

The monitor is trusted to collect the traffic reports and provide proper access to them, while networks may be honest (report true packet counts) or dishonest (report fake packet counts in an effort to exaggerate their performance). Like prior work, our transparency protocol by design prevents dishonest networks from exaggerating their performance. For example, consider Fig. 5.1 and an aggregate A that crosses witnesses X_1, X_2, and Y_1. Suppose ISP X drops a packet from A but wants to hide this fact. To this end, it makes X_2 report a fake packet count for A (increased by one relative to the true packet count). As a result, the monitor thinks that the packet was lost on

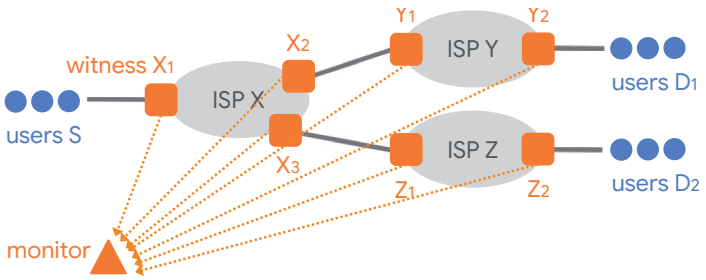


Figure 5.1 – Transparency introduces global adversary.

the inter-domain link between X_2 and Y_1 and attributes the loss to *both* ISPs X and Y. Hence, by being dishonest, ISP X not only cannot hide its true packet loss, but it also falsely attributes packet loss to neighbor ISP Y, causing a conflict with that neighbor [20].

Moreover, we consider an adversary who is passive and aims to "trace" a target flow f: given a set of aggregates $\mathscr{A}$, one of which contains f, she wants to determine which aggregate is most likely to contain f. She has the following information: (a) f's pattern; (b) all the traffic reports for all the aggregates in $\mathscr{A}$ published within the observation window W. This adversary could be, for example, a government, who has subpoenaed an ISP (to gain access to a user's Internet connection), as well as the monitor (to obtain the traffic reports).

We illustrate how this relates to low-latency anonymity networks through an example: Suppose user S in Fig. 5.1 sends a flow f to some destination through Tor. Consider an adversary, Eve, who monitors S's Internet connection and learns f's packet-arrival pattern. Moreover, Eve knows that f is observed at witness X_1 and then either witness X_2 or X_3. Without any extra information, Eve cannot determine f's destination. However, if she obtains the traffic reports published by witnesses X_2 and X_3, she can extract the patterns of the aggregates observed at these two witnesses, correlate them with f's pattern, and guess which one of these aggregates contains f. By repeating this process, she can guess the sequence of witnesses that observe f. If she guesses correctly, she has narrowed down the flow's destination within an IP prefix. In some cases, this is all Eve needs to know, e.g., if the IP prefix belongs to a censored content provider.

It has already been shown that low-latency anonymity networks are vulnerable to similar adversaries. In one case, the adversary observes a target flow f and a set of aggregates $\{A_i\}$, one of which contains f; her goal is to guess which A_i contains f [26]. In another case, the adversary observes a random sample of a target flow f and a set of packet sequences, one of which is also a random sample of f; her goal is to guess which packet sequence is a random sample of f [60]. Our adversary is similar to these, in that she knows the pattern of the target flow, and she also has information about other traffic, observed at different points in the network, which may be correlated with the target flow.

The new aspect of our adversary is that she learns about traffic (other than the target flow) from transparency reports. We are interested in this particular adversary, because we want to assess the

new anonymity risk that a transparency protocol would introduce. Our adversary would pose no threat to an anonymity network that explicitly makes aggregates indistinguishable by introducing latency and/or fake traffic [50]. We do not consider active adversaries [34] or software exploits that target anonymity-network browsers.[1]

5.2.3 Problem Statement

We want a transparency protocol that strikes a good balance between being useful (enabling accurate assessment of network performance) and obstructing flow tracing. Given that our adversary has access to the traffic reports published by the witnesses, any obfuscation of information must happen at the witnesses, before the reports are published.

Hence, we look for an algorithm that takes as input a traffic report and modifies it, such that the report makes flow tracing "as hard as possible" while meeting a target "utility level." Each witness can then apply this algorithm to each traffic report it produces. We assume that each witness has only a local view (the traffic it observes and the reports it produces), but not the adversary's global view (the reports produced by other witnesses). Moreover, from the witness's point of view, any flow contained in an aggregate could be a target flow, which means that the algorithm cannot focus on making only specific flows hard to trace.

We put "as hard as possible" and "utility level" in quotes, because they are not meaningful until we define metrics that capture how much a report helps/obstructs flow tracing versus how much it helps transparency. Defining metrics was a key part of our work, and we state them later in the chapter. In the next subsection, we summarize the most relevant anonymity metrics from the literature that we used as basis and inspiration.

5.2.4 Anonymity Metrics

In general, anonymity metrics characterize an adversary's uncertainty about linking an item of interest to a target user's identity [71]. In our context, the item of interest is the target flow while the target user's identity is an aggregate observed at a witness.

Traceability [26]. This metric is useful when one knows the packet-arrival patterns of a target flow f and two candidate aggregates, A_0 and A_1, one of which contains f; her goal is to trace f (decide which aggregate is more likely to contain it). Traceability is defined as:

$$TR \triangleq \log \frac{\mathcal{L}\{H_0\}}{\mathcal{L}\{H_1\}}, \tag{5.2}$$

where $\mathcal{L}$ denotes likelihood, and H_0 (H_1) is the hypothesis that A_0 (A_1) contains f. Traceability 0 means that the two hypotheses are equally likely, i.e., the packet-arrival patterns do not help trace f. The absolute value of traceability indicates the difference between the log-likelihoods of

[1] FBI Used Firefox Exploit to Shutdown Illegal Site Running on Tor Network

the two hypotheses, so, the larger it is, the more the packet-arrival patterns help trace the flow.

The paper that introduced traceability showed how to compute it assuming independent packet arrivals that follow a Poisson distribution for the target flow and a uniform distribution for all other traffic. Indeed, when we experimented with synthetic flows generated from a Poisson model, traceability worked, i.e., our adversary could use it to trace target flows. However, when we experimented with flows extracted from CAIDA traffic traces, the flows' packet-arrival patterns were not always Poisson, and computing traceability under the assumption that they were did not work. For instance, it would be the case that a target flow's packet-arrival pattern was clearly visible within one of the candidate aggregates, and yet traceability was close to 0.

Cross-correlation [74]. This metric captures the similarity between the packet-arrival pattern of a target flow f and a traffic report[2] $R(A)$ during a time interval $[t_1, t_2]$. It is defined as:

$$CC(f, R(A), [t_1, t_2]) \triangleq \sum_{t \in [t_1, t_2]} (N_A(t) - \lambda_A)(N_f(t) - \lambda_f). \tag{5.3}$$

In our context, cross-correlation is more practical than traceability, because it does not require any assumptions about packet arrivals. Moreover, when we experimented with flows extracted from CAIDA traffic traces, cross-correlation worked, i.e., it did enable our adversary to trace target flows.

However, neither metric captures intuitively our adversary's power. For instance, it is not clear what values of traceability or cross-correlation we should target in order to argue that our adversary does not pose a threat to anonymity networks.

5.3 Approach

In this section, we describe first the metric that we use to capture our adversary's uncertainty (§5.3.1), then the measurements that motivated our approach (§5.3.2), and then the idea of using coarser time granularity for anonymization (§5.3.3).

5.3.1 Metric: T-Anonymity Set Size

Our adversary wants to trace target flow f across a set of candidate aggregates $\mathscr{A} = \{A_1, A_2, \ldots A_{|\mathscr{A}|}\}$. The ground truth is that aggregate A_x contains f. The adversary computes a likelihood distribution $\mathscr{L} = \{l_1, l_2, \ldots l_{|\mathscr{A}|}\}$, where l_i is her estimated likelihood that A_i contains f.

We want a metric that captures the adversary's uncertainty in tracing the target flow to the correct aggregate, akin to an anonymity set size. In particular, we want our metric to have value:

[2]The original definition is for a flow and an aggregate. We define it for a flow and a traffic report, because, in our context, the adversary learns about the aggregate from a traffic report.

$$\begin{cases} 1, & \text{if } l_x = 1 \\ \in (1, |\mathscr{A}|), & \text{if } l_x \in \left(\frac{1}{|\mathscr{A}|}, 1\right) \\ |\mathscr{A}|, & \text{if } l_x \in \left[0, \frac{1}{|\mathscr{A}|}\right] \end{cases}$$

The first row describes the scenario where the adversary knows the 1 aggregate that contains f. The last row describes the scenario where the adversary either has no information (she believes that all $|\mathscr{A}|$ aggregates contain f with the same likelihood $\frac{1}{|\mathscr{A}|}$), or has misleading information (she believes that aggregate A_x contains f with likelihood $< \frac{1}{|\mathscr{A}|}$). The middle row describes all other scenarios, where the adversary has some correct information.

At first we defined our adversary's anonymity set size as $2^{H(\mathscr{L})}$, where H denotes entropy, with the rationale that entropy is the standard way to quantify the uncertainty resulting from a likelihood distribution. This metric, however, does not work when the adversary has misleading information. For example, consider two scenarios: (a) the adversary is certain that aggregate A_x contains the target flow f, which is true; (b) the adversary is certain that aggregate A_y contains f, which is false. In both scenarios the entropy of her likelihood distribution is $H(\mathscr{L}) = 0$, and $2^{H(\mathscr{L})} = 1$, which indicates that the adversary has traced f to 1 aggregate, but ignores that, in scenario (b), the tracing is false.

Instead, we define our adversary's **T-anonymity set size** as:

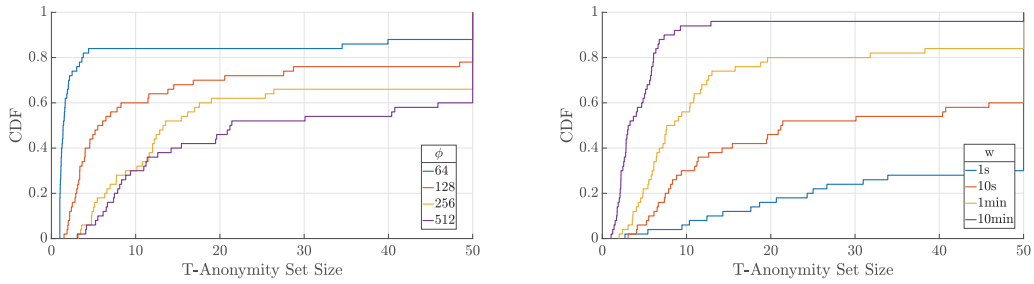
$$\mathscr{S} \triangleq \min\left\{\frac{1}{l_x}, |\mathscr{A}|\right\}. \tag{5.4}$$

Recall that: $\mathscr{A}$ is the set of candidate aggregates, the ground truth is that aggregate A_x contains the target flow f, and l_x is the adversary's estimated likelihood that A_x contains f. This metric satisfies our requirements and has an intuitive meaning. For instance, suppose the adversary knows that either A_x or A_y contain f, but has no further information; then, her T-anonymity set size is $\min\left\{\frac{1}{0.5}, |\mathscr{A}|\right\} = 2$, which indicates that she has correctly traced f to one of 2 aggregates. However, if the adversary believes that A_x contains f with likelihood 0.9, while A_y contains f with likelihood 0.1, then her T-anonymity set size is $\min\left\{\frac{1}{0.9}, |\mathscr{A}|\right\} \approx 1.12$, which indicates that she has correctly traced f to almost 1 aggregate.

Our metric is related to prior work as follows: Consider a slightly different context from ours, where: there is an item of interest, associated with one user from a set $\mathscr{A} = \{A_1, A_2, \ldots A_{|\mathscr{A}|}\}$; a true distribution $\mathscr{P} = \{p_1, p_2, \ldots p_{|\mathscr{A}|}\}$, where p_i is the likelihood that the item is associated with user A_i; and an estimated distribution $\mathscr{L} = \{l_1, l_2, \ldots l_{|\mathscr{A}|}\}$, which is an adversary's estimate of $\mathscr{P}$. To quantify the adversary's uncertainty about the true distribution, it has been proposed to use the **relative entropy** or **KL divergence** from $\mathscr{P}$ to $\mathscr{L}$:

$$D_{\mathrm{KL}}(\mathscr{P} \| \mathscr{L}) \triangleq \sum_i p_i \log_2 \frac{p_i}{l_i}. \tag{5.5}$$

In our context (where the ground truth is that A_x contains the target flow f), we could say that



(a) Observation window size $w = 10$sec, varying number of flows per aggregate ϕ.

(b) Number of flows per aggregate $\phi = 512$, varying observation window size w.

Figure 5.2 – CDF of the adversary's T-anonymity set size as a function of flows per aggregate (left) and observation window (right).

the true likelihood distribution is $\mathscr{P} = \{p_i | p_{i=x} = 1, p_{i\neq x} = 0\}$, hence $D_{\mathrm{KL}}(\mathscr{P} \| \mathscr{L}) = p_x \log_2 \frac{p_x}{l_x} = \log_2 \frac{1}{l_x}$. The standard way to convert this entropy to an anonymity set size would be $2^{\log_2 \frac{1}{l_x}} = \frac{1}{l_x}$. Hence, one can view our metric as the anonymity set size that corresponds to the relative entropy between the ground truth and our adversary's estimated likelihood distribution.

There remains the question of how to compute our adversary's likelihood distribution $\mathscr{L}$. First, the target flow f cannot belong to an aggregate A_i if f has more packets than A_i during any time tick t in the observation window W. Hence:

$$l_i \triangleq 0, \quad \textit{if } \exists t \in W, s.t. f(t) > N_{A_i}(t). \tag{5.6}$$

Otherwise, we compute l_i based on the cross-correlation between f and $R(A_i)$:

$$l_i \triangleq \frac{CC^+\left(f, R(A_i), W\right)}{\sum_{\forall A_j \in \mathscr{A}} CC^+\left(f, R(A_j), W\right)}, \tag{5.7}$$

where

$$CC^+\left(f, R(A), [t_1, t_2]\right) \triangleq \max\left\{ CC\left(f, R(A), [t_1, t_2]\right), 0 \right\}. \tag{5.8}$$

We count only positive cross-correlation between f and $R(A)$ as an indication that f belongs to A. We did experiment with an alternative approach, where we computed l_i as a normalized version of $|CC(f, R(A_i), W)|$, but we found that our adversary drew slightly worse conclusions that way. This is because we assume that our adversary correctly aligns target flow patterns to aggregate reports. As a result, negative cross-correlation between f and $R(A)$ is actually an indication that a flow does *not* belong to A.

5.3.2 Would Transparency Affect Anonymity?

We use Internet backbone traces made available by CAIDA, collected at the equinix-nyc monitor, direction A, from March to November 2018. From each trace, we extract TCP and UDP flows;

then we create aggregates by grouping flows together. We assume a time tick of 1msec (i.e., a traffic report contains an aggregate's packet count every 1msec). In each experiment, we emulate the scenario where: an adversary obtains traffic reports for 50 aggregates, $A_1, A_2, \ldots A_{50}$ and wants to trace 50 target flows, $f_1, f_2, \ldots f_{50}$; the ground truth is that aggregate A_i contains flow f_i. The number of flows per aggregate, ϕ, and the adversary's observation window, w, vary per experiment.

In the experiments we present, each target flow contributes a maximum of $\rho = 1$ packet during any given time tick (hence maximum rate 1.5Mbps). In general, burstier, higher-rate flows are easier to trace. Low-latency anonymity networks do not aim to protect flows of arbitrary rate, for example, Tor's hidden-service statistics aim to protect flows that contribute up to 1MiB over 24 hours [41]. In our context, as stated above, a target flow f cannot belong to a candidate aggregate A if f has more packets than A during any time tick t in the observation window. The larger f's bursts, the more candidate aggregates the adversary can exclude with this rationale. Hence, we think the interesting question is whether transparency would affect the anonymity of relatively low-rate flows, which could not be trivially traced due to their bursts. For this reason, we cropped our target flows, such that each contributes up to $\rho = 1$ packet during any given time tick. To satisfy this constraint, from the $50 \times 600\text{k}$[3] flow contributions per time tick that we considered, we had to crop 1%.

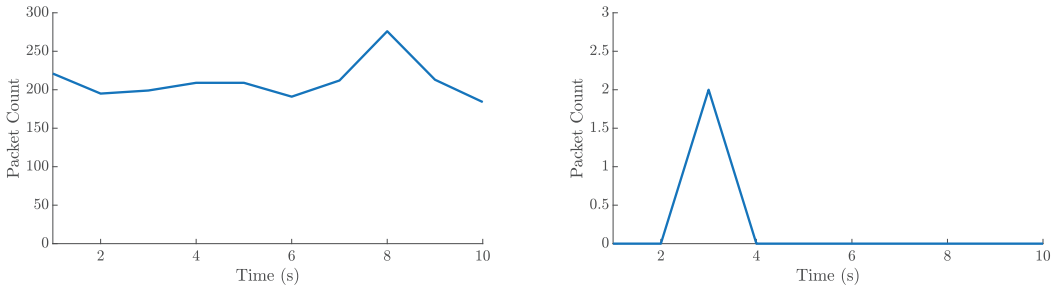
First, we look at how the adversary's uncertainty depends on the number of flows per aggregate ϕ. Recall that, in our context, an aggregate is all traffic observed at a witness with a unique source and destination IP prefix pair. So, we expect most aggregates to contain at least hundreds of flows; however, it could happen that, for a short time window, an aggregate contains fewer active flows than usual, potentially making these flows more vulnerable to tracing. The question then is: what can our adversary do when ϕ is in the hundreds, and what can she do when ϕ drops to tens of flows per aggregate?

Fig. 5.2a shows the cumulative distribution function (CDF) of the adversary's T-anonymity set size $\mathscr{S}$ when her observation window size is $w = 10$sec, while the number of flows per aggregate ϕ varies. When $\phi = 512$, the adversary traces no target flow to a unique aggregate, but she still traces 6% of the target flows to <5 candidate aggregates. When $\phi = 64$, she traces 64% of the target flows to a unique aggregate, and 84% of the target flows to <5 candidate aggregates.

Next, we look at how the adversary's uncertainty depends on her observation window size w. We expect that the adversary's uncertainty drops as w increases, and the question is how fast.

Fig. 5.2b shows the CDF of the adversary's T-anonymity set size $\mathscr{S}$, when there are $\phi = 512$ flows per aggregate, while the adversary's observation window size varies. As we saw in the previous graph, when $w = 10$sec, the adversary traces no target flow to a unique aggregate, but she still traces 6% of the target flows to <5 candidate aggregates. When $w = 10$min, she traces 16% of the target flows to a unique aggregate, and 62% of the target flows to <5 candidate aggregates.

[3] The maximum observation window we considered is 10min = 600k time ticks.



(a) A packet-arrival pattern that is easy to trace. (b) A packet-arrival pattern that is hard to trace.

Figure 5.3 – Examples of actual packet flows that are easy (left) and hard (right) to trace.

As expected, the adversary's uncertainty increases as ϕ gets bigger and w gets smaller ($\mathscr{S}$'s CDF shifts to the right); however, she always manages to trace a few target flows to a relatively small number of candidate aggregates. For example, even when $\phi = 512$ flows per aggregate, and the adversary's observation window is barely $w = 1$sec (rightmost curve in Fig. 5.2b), there is still one target flow for which $\mathscr{S} < 5$. As expected, these are flows with peculiar packet-arrival patterns, e.g., bursts of unusual duration or period, that make them stand out even within 511 other flows. For example, the flow shown in Fig. 5.3a is vulnerable to tracing, because it is the only flow in our dataset to contribute such a high number of packets within a few seconds. In contrast, the flow shown in Fig. 5.3b is hard to trace, because it only contributes two packets in total—a pattern that is easily hidden within an aggregate.

5.3.3 Coarser Time Granularity as Noise

The first approach we tried was differential privacy [32]: Consider a witness that observes two aggregates, A_1 and A_2, that differ in a single flow f; and publishes traffic reports $R(A_1)$ and $R(A_2)$. Ideally, if $R(A_1)$ contains a tuple $\langle t, N_{A_1}(t)\rangle$, and $R(A_2)$ contains a tuple $\langle t, N_{A_2}(t)\rangle$, it should be the case that probabilities of $N_{A_1}(t)$ and $N_{A_2}(t)$ taking any arbitrary value N are approximately equal. If we could guarantee this property for any A_1, A_2, and f, then we could say that the transparency protocol is "differentially private," in the sense that traffic reports never reveal any information that could help our adversary trace a target flow.

We explored this approach but could not guarantee any meaningful differential privacy without making the traffic reports useless: We tried adding noise to the packet counts of a traffic report so as to guarantee ϵ-differential privacy; we considered both standard Laplace noise and a more recent Fourier-based variant [73]. When we set ϵ to any typical value (e.g., $\in (0,1)$), the modified packet counts were so noisy that any statistic computed from them was arbitrarily unreliable. When we bounded the difference between the original and modified packet counts to any reasonable value (e.g., relative error 10%), the ϵ for which we could guarantee ϵ-differential privacy was so large (tens of thousands) that we could not reason about its meaning any more.

In retrospect, given the purpose of a transparency protocol, additive noise does not make sense

as a first counter-measure (although it could be useful as an enhancement): If a traffic report contains packet counts at a granularity of 10msec, and we decide that that reveals too much information, it does not make sense to add noise to the packet counts while keeping the same, fine time granularity.

Hence, we explore the following idea: instead of adding noise to the packet counts of traffic reports, we coarsen their time granularity. This is another form of noise, and it has two benefits:

(1) It allows us to control report utility: Suppose traffic reports have time granularity τ, i.e., each witness publishes a packet count per aggregate every τ time ticks. By coarsening the time granularity of the reports (increasing τ), we do not make them less reliable: if the packet counts are perfectly accurate, the packet-loss rates computed from them will also be perfectly accurate, albeit averaged over longer time intervals.

(2) It allows us to preserve the incentive structure of the transparency protocol (§5.2.2): By coarsening the time granularity of the reports, we do not change the fact that reports are expected to contain exact per-aggregate packet counts. Hence, as long as we can align the reports produced by subsequent witnesses for the same aggregate, a network cannot escape the blame for a lost packet—it can only shift it from an internal path to one of its own inter-domain links, which does not improve its perceived performance and causes a conflict with a neighboring network.

5.4 Algorithm

We now present our algorithm: first an overview (§5.4.1), then an "idealized" version (§5.4.2), and then a more practical online version (§5.4.3).

5.4.1 Overview

Given an aggregate A and an observation window W, our algorithm takes as input:

- A traffic report[4]

$$R(A) \triangleq \{ \langle t, N_A(t) \rangle, \forall t \in W \}. \tag{5.9}$$

- The patterns of A's flows

$$\{ \langle t, N_f(t) \rangle, \forall t \in W, \forall f \in A \}. \tag{5.10}$$

[4]Technically, this traffic report can be reconstructed from the second input (the patterns of A's flows). We state it as a separate input because we think that helps make the algorithm description clearer.

It produces as output a traffic report $R_o(A)$, which contains packet counts not per time tick (as the input report), but per time bin:

$$R_o(A) \triangleq \{ \langle T, N_A(T) \rangle, \forall T \in \mathcal{T} \}, \tag{5.11}$$

where $\mathcal{T}$ is a set of consecutive, non-overlapping time bins that cover the observation window W.

Our algorithm takes the following configuration parameters:

- The maximum flow burst size ρ. Flows with bigger bursts are easier to trace: The adversary knows that a target flow f does not belong to a candidate aggregate A_i if she knows that f has more packets than A_i during any given time interval. The burstier f is, the more aggregates the adversary can exclude with this rationale. Our algorithm tries to protect flows that contribute up to ρ packets during any single time tick.
- The maximum bin size τ. Our algorithm produces time bins that contain up to this many time ticks. This parameter is our way of ensuring that the output report retains a certain utility level.

To capture the similarity between a target flow f and the output traffic report $R_o(A)$, we define the cross-correlation between f and $R_o(A)$ and its positive-only version as:

$$CC\big(f, R_o(A), [t_1, t_2]\big) \triangleq \sum_{t \in [t_1, t_2]} \left(\frac{N_A(T_t)}{||T_t||} - \lambda_A \right) (N_f(t) - \lambda_f), \tag{5.12}$$

$$CC^+\big(f, R_o(A), [t_1, t_2]\big) \triangleq \max\big\{ CC\big(f, R_o(A), [t_1, t_2]\big), 0 \big\}. \tag{5.13}$$

where $T_t \in \mathcal{T}$ is the time bin that contains time tick t. This is essentially the same definition as in Eq. (5.3), with the difference that $N_A(t)$ has been replaced by $\frac{N_A(T_t)}{||T_t||}$, because the adversary does not see the original packet counts $N_A(t)$ any more, but only the modified, coarser packet counts $N_A(T_t)$.

We do not try to design an optimal algorithm: Recall that our adversary collects traffic reports for a set of candidate aggregates observed at different witnesses, and tries to trace a target flow f; the metric for her success is her T-anonymity size $\mathcal{S}$ for f. An optimal algorithm would either maximize $\mathcal{S}$ subject to some minimum report utility; or maximize report utility subject to some minimum $\mathcal{S}$. Either approach requires knowledge of f's pattern as well as the patterns of all the candidate aggregates; whereas our algorithm runs at a single witness and takes as input only $R(A)$ and the patterns of A's flows.

The simplest solution would be to choose time bins of fixed duration τ; as we show in our evaluation, this protects most realistic flows, but has two disadvantages: First, it introduces unnecessary noise, because it coarsens the report's time granularity uniformly, even during time

intervals when the report does not improve the adversary's knowledge. Second, it can be bad for flows whose pattern happens to align in an unlucky way with the time bins. For instance, consider a flow that is active for a window of x time ticks, then inactive for a window of x time ticks, and the pattern repeats; if each time bin happens to align with an active or inactive window, then coarsening time granularity from 1 to x time ticks may not increase at all the adversary's uncertainty (depending on the other traffic). Such "on-off" flows do exist, albeit rarely, in the CAIDA traces.

Our solution relies on the concept of a **virtual flow** v: given an output traffic report $R_o(A)$, v is a synthetic flow that, during any time bin T, has the same pattern as the real flow that has the highest cross-correlation with $R_o(A)$ during time bin T. More precisely,

$$N_v(t) \triangleq N_{f_T}(t),\ \forall\, t \in T, \tag{5.14}$$

where

$$f_T \triangleq \arg\max_f CC^+(f, R_o(A), T). \tag{5.15}$$

Our algorithm tries to choose the time bins of the output traffic report, such that the report reveals as little information as possible about v. The rationale is that v consists of the most vulnerable pieces of A's real flows; so, if the output report hides v's pattern, we expect that it will also hide the patterns of A's real flows.

5.4.2 Idealized Algorithm

We first designed an idealized version of our algorithm (that we call MorphIT$_{id}$), which finds the time bins $\mathcal{T}$ that cover the observation window W while minimizing the following metric:

$$\sum_{\forall T \in \mathcal{T}} \max_f CC^+(f, R_o(A), T). \tag{5.16}$$

This is simply the positive-only cross-correlation (Eq. (5.13)) between the virtual flow v (Eq. (5.15)) and the output traffic report $R_o(A)$ (Eq. (5.11)) during the observation window W.

Our algorithm relies on dynamic programming and defines three matrices:

$S[j+1, i]$ specifies how much our optimization metric (Eq. (5.16)) will increase if the output report already covers time interval $[0, j]$ and we add time bin $[j+1, i]$:

$$S[j+1,i] = \begin{cases} \infty, & \textit{if } i-j > \tau \textit{ OR } N_A\big([j+1,i]\big) < (i-j)\rho, \\ \max_f CC^+(f, R_o(A), [j+1,i]), & \textit{otherwise} \end{cases} \tag{5.17}$$

Notice that $S[j+1, i] = \infty$ if time bin $[j+1, i]$ exceeds the maximum bin size τ, or if aggregate A

contributes fewer than $(i-j)\rho$ packets in time bin $[j+1, i]$ (in which case we could not protect flows with such burst sizes).

$S_{opt}[k, i]$ keeps track of the minimum value of the optimization metric when the output report covers time interval $[1, i]$ divided in k time bins:

$$S_{opt}[k,i] = \begin{cases} S[1,i], & \text{if } k=1, \\ S_{opt}[k-1, j^*] + S[j^*+1, i], & \text{if } k>1, \end{cases} \tag{5.18}$$

$$j^* = \underset{k-1 \le j \le i-1}{\operatorname{argmin}} \left\{ S_{opt}[k-1, j] + S[j+1, i] \right\}. \tag{5.19}$$

$T_{opt}[k, i]$ keeps track of the time bins that lead to $S_{opt}[k, i]$ (in particular, it specifies the beginning of the last time bin that leads to $S_{opt}[k, i]$).

Once the algorithm has filled S_{opt} and T_{opt}, it picks the time bins that lead to $S_{opt}[1, w]$ by backtracking from $T_{opt}[k^*, w]$, where:

$$k^* = \underset{\lceil w/\tau \rceil \le k \le w}{\operatorname{argmin}} \left\{ S_{opt}[k, w] \right\}.$$

The complexity of the idealized algorithm is

$$\mathcal{O}(w^3) + \mathcal{O}(\phi w^2),$$

where w is the size of the observation window covered by the input traffic report, and ϕ is the number of flows in aggregate A. The first term comes from filling S_{opt}: to fill each position, the algorithm examines $\tau = \mathcal{O}(w)$ entries; this is done for each of the $w \times w$ positions of the matrix, leading to $\mathcal{O}(w^3)$. The second term comes from filling S: to fill $S[j, i]$, the algorithm computes $CC^+(f, R_o(A), [j, i])$ for each of the ϕ participating flows; this is done for each of the $w \times w$ positions of the matrix, leading to $\mathcal{O}(\phi w^2)$.

Given that we want witnesses to run our algorithm in real time, this complexity becomes prohibitive when w extends beyond a few tens of seconds.

5.4.3 Online Algorithm

To make our algorithm practical, we designed an "online" version (that we call MorphIT$_\omega$), which divides the observation window into smaller windows of size ω, applies the idealized algorithm to each of them, and combines all the resulting outputs into one that covers the entire observation window.

Intuitively, as the size of the active window ω approaches the size of the observation window w, the performance of the online version improves and approaches that of the idealized one;

however, we cannot guarantee that the performance gap between the two closes smoothly as ω increases. Fig. 5.4 illustrates an extreme scenario where any $\omega < w$ yields bad results: Aggregate A_x has packets only during the first time tick, while aggregate A_y has packets only during the last time tick w. If the adversary is trying to trace a flow to one of these two aggregates, she will clearly succeed, unless the output traffic reports consist of a single time bin of length w—in which case the two aggregates become indistinguishable. In this scenario, any $\tau < w$ and any $\omega < w$ would have no impact on the adversary's T-anonymity set size.

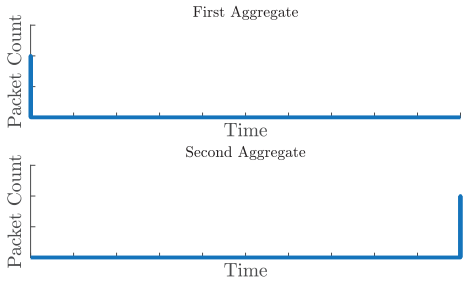


Figure 5.4 – Example of two aggregates that require $\tau = \omega = w$.

The question is: are there values of ω that make the algorithm deployable while maintaining most of the benefit of the idealized algorithm? The answer depends on flow duration and patterns, and we study it in our evaluation section.

5.5 Experimental Evaluation

After describing our setup (§5.5.1), we present our algorithm's performance (§5.5.2) and compare it to a simpler alternative (§5.5.3) and a state-of-the-art anonymization tool based on differential privacy (§5.5.4). We discuss processing overhead in App. §C.

5.5.1 Setup

The basic experimental setup is the same as in §5.3.2: Each time tick lasts 1msec. In each experiment, we consider: 50 aggregates, $A_1, A_2, \ldots A_{50}$, and an adversary who wants to trace 50 target flows, $f_1, f_2, \ldots f_{50}$; the ground truth is that aggregate A_i contains flow f_i. The number of flows per aggregate ϕ and the adversary's observation window size w vary per experiment.

We experiment with three types of traffic:

1. **Poisson**: We generate flows with Poisson packet arrivals of average rate $\lambda_f = 1$ packet per time tick. We create an aggregate by grouping ϕ of these flows. This is similar to the experimental setup used in [26], the work that is closest in spirit to ours. There is also evidence that real traffic flows can have Poisson behavior [49].

2. **Real**: We extract TCP and UDP flows from Internet backbone traces as stated in §5.3.2. We create an aggregate by grouping ϕ randomly chosen flows. We crop the target flows such that each of them contributes up to $\rho = 1$ packet during any given time tick. So, we use the traces to obtain realistic flows, but assume that an aggregate may consist of any random subset of these flows; we do not rely on the traces to draw any conclusion about the nature of aggregates.

3. **On-off**: We craft a target flow with 10 packets at time tick 31, 10 packets at time tick 32, and no other traffic. Then we craft another target flow with the same pattern, but shifted by 2msec, i.e., 10 packets at time tick 33, 10 packets at time tick 34, and no other traffic. We craft 24 more such pairs of target flows, where each flow has traffic during only 2 time ticks, and one flow is shifted by 2msec relative to the other. We create an aggregate by grouping 1 on-off flow with $\phi - 1$ real flows (extracted from traces). The purpose of this traffic pattern is to illustrate the limitations of the Uniform algorithm.

We consider the following algorithms:

1. **MorphIT**$_{100}$: This is the online version of our algorithm with an active window of $\omega =$ 100msec.

2. **MorphIT**$_{id}$: This is the idealized version of our algorithm. We run it whenever possible (when the observation window is small enough for the algorithm to finish in reasonable time), to give a sense of how much better it performs than our online algorithm.

3. **Uniform**: This is a simpler alternative (§5.4.1) that always picks fixed time bins of equal size τ. If it works well, then there is no reason for a more complex algorithm like MorphIT.

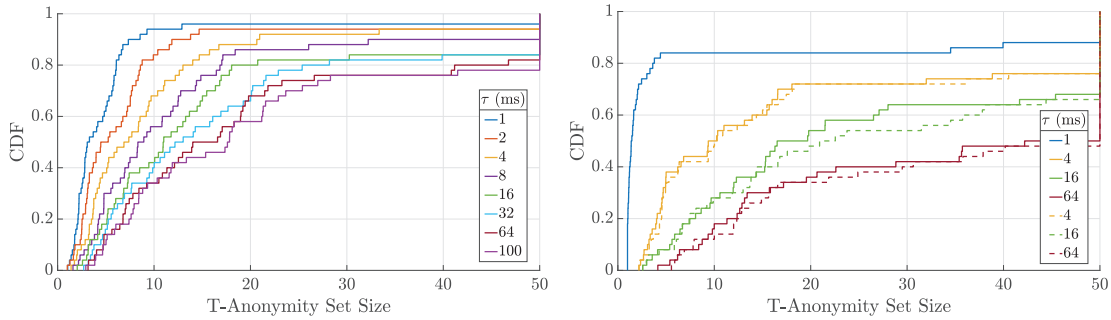
4. **PrivCount** [47]: This is a state-of-the-art system that uses Gaussian noise to provide (ϵ, δ)-differential privacy [31] for a number of statistics aggregated across the Tor network and over time (e.g., number of TCP connections exiting Tor within 24 hours).

We adjust the computation of the adversary's likelihood distribution $\mathscr{L}$ in a straightforward manner: Consider a target flow f, a candidate aggregate A_i, and the time bins $\mathscr{T}_i$ from A_i's report $R_o(A_i)$. First, f cannot belong to A_i if, during any one time bin in $\mathscr{T}$, f has more packets than A_i:

$$l_i \triangleq 0, \quad if\ \exists T \in \mathscr{T}, s.t.\ f(T) > N_{A_i}(T). \tag{5.20}$$

Otherwise, we compute l_i based on the cross-correlation between f and $R_o(A_i)$:

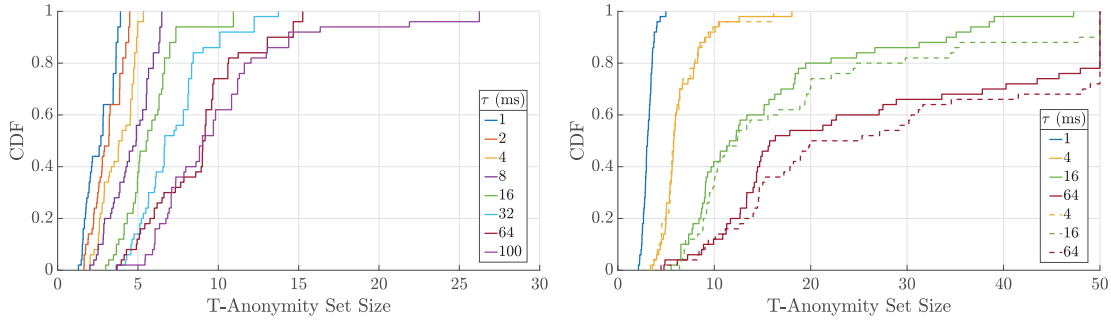
$$l_i \triangleq \frac{CC^+\left(f, R_o(A_i), W\right)}{\sum_{\forall A_j \in \mathscr{A}} CC^+\left(f, R_o(A_j), W\right)}. \tag{5.21}$$



(a) "Long observation" scenario: $\phi = 512$ flows/aggregate, $w = 10$min.

(b) "Sparse aggregates" scenario: $\phi = 64$ flows/aggregate, $w = 10$sec.

Figure 5.5 – CDF of the adversary's T-anonymity set size given real flows. The solid curves are achieved by MorphIT$_{100}$, while the dotted curves are achieved by MorphIT$_{id}$. The max bin size τ varies.



(a) "Long observation" scenario: $\phi = 512$ flows/aggregate, $w = 10$min.

(b) "Sparse aggregates" scenario: $\phi = 64$ flows/aggregate, $w = 10$sec.

Figure 5.6 – CDF of the adversary's T-anonymity set size given Poisson flows. The solid curves are achieved by MorphIT$_{100}$, while the dotted curves are achieved by MorphIT$_{id}$. The max bin size τ varies.

5.5.2 MorphIT Performance

We consider two scenarios from §5.3.2 where the adversary had little uncertainty:

1. **Long observation**: There are $\phi = 512$ flows per aggregate, and the adversary's observation window is $w = 10$min.
2. **Sparse aggregates**: There are only $\phi = 64$ flows per aggregate, and the adversary's observation window is $w = 10$sec.

Fig. 5.5 shows the CDF of the adversary's T-anonymity set size $\mathscr{S}$ given real flows, in the "long observation" scenario (5.5a) and in the "sparse aggregates" scenario (5.5b). The solid curves were obtained with MorphIT$_{100}$, while the dotted curves were obtained with MorphIT$_{id}$ (we do

not show MorphIT$_{id}$'s performance for $w = 10$min, because the algorithm is infeasible for such a large w). Different curves correspond to different max bin sizes τ.

Consider Fig. 5.5a. Without any modification to the traffic report ($\tau = 1$msec, leftmost curve), the adversary traces 16% of the target flows to a unique aggregate and 62% of the target flows to <5 candidate aggregates. As τ increases, we allow our algorithm to modify more the traffic report, and the adversary's uncertainty increases ($\mathscr{S}$'s CDF shifts to the right). For $\tau = 16$msec, the adversary traces no target flow to a unique aggregate; for $\tau = 64$msec, she traces 14% of the target flows to a set of <5 candidate aggregates. Regarding report utility, as long as $\tau < 1$sec, users of the transparency protocol can still use the reports to compute network performance at a sub-second time granularity.

Consider Fig. 5.5b. Recall that this is a particularly adversarial scenario, with only $\phi = 64$ flows per aggregate. Without any modification to the traffic report ($\tau = 1$msec, leftmost curve), the adversary traces 66% of the target flows to a unique aggregate and 84% of the target flows to <5 candidate aggregates. For $\tau = 64$msec, she traces no target flow to a unique aggregate and 1 target flow to <5 candidate aggregates. Regarding the relative performance of the two algorithms, MorphIT$_{100}$ closely follows the T-anonymity set sizes achieved by MorphIT$_{id}$.

Fig. 5.6 shows the CDF of the adversary's T-anonymity set size $\mathscr{S}$ given Poisson flows, in the "long observation" scenario (Fig. 5.6b) and in the "sparse aggregates" scenario (Fig. 5.6a).

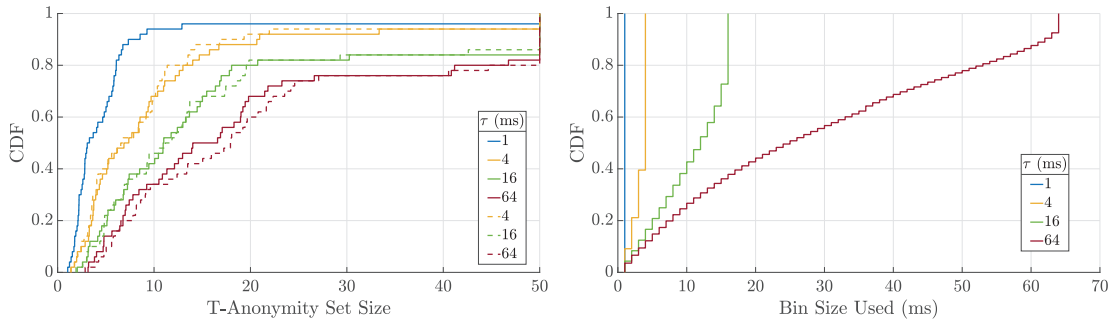
Comparing our results given Poisson versus real flows: The adversary's uncertainty is, in general, lower with Poisson flows. We think that this is due to the fact that all Poisson flows are active throughout the observation window, whereas real flows are typically active for significantly shorter time intervals. On the other hand, the adversary's uncertainty is more stable with Poisson flows ($\mathscr{S}$'s CDF is closer to vertical), which is not surprising given that their packet arrivals follow the same distribution.

5.5.3 Comparison to Uniform

We expected Uniform to achieve adversary uncertainty similar to MorphIT$_{id}$, but introduce a significant amount of unnecessary noise (because it coarsens the entire traffic report to time granularity τ, whether that helps anonymity or not).

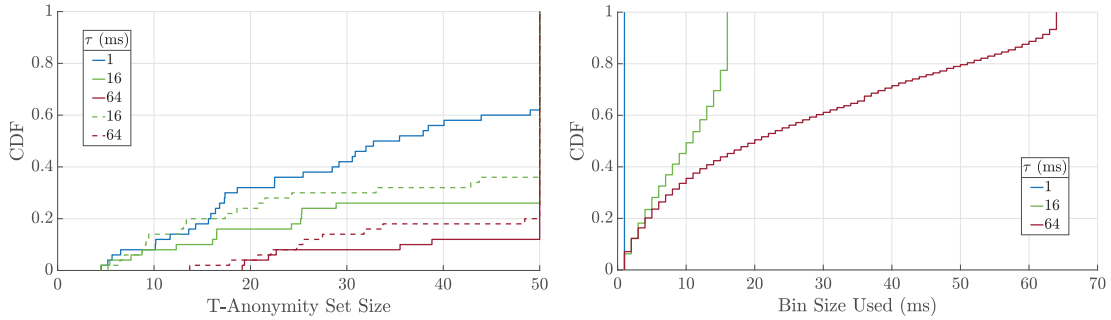
Fig. 5.7 compares Uniform to MorphIT$_{100}$ in the "Long observation" scenario, given real flows. Uniform achieves similar adversary uncertainty (Fig. 5.7a), but introduces significantly more noise: Fig. 5.7b shows the CDF of the bin sizes picked by MorphIT$_{100}$ in each scenario. We see that it resorts to the max bin size sparingly, especially for the larger values of τ. For instance, when $\tau = 64$msec, 50% of the bins have size <25msec, while 80% of the bins have size <53msec.

Fig. 5.8 compares Uniform to MorphIT$_{100}$ in the "Sparse aggregates" scenario, given on-off target flows. Uniform achieves slightly worse (lower) adversary uncertainty (Fig. 5.8a), while it



(a) CDF of the adversary's T-anonymity set size. (b) CDF of the bin sizes picked by MorphIT$_{100}$.

Figure 5.7 – MorphIT$_{100}$ (solid curves) versus Uniform (dotted curves) performance given real flows. "Long observation" scenario: $\phi = 512$ flows/aggregate, $w = 10$min. The max bin size τ varies.



(a) CDF of the adversary's T-anonymity set size. (b) CDF of the bin sizes picked by MorphIT$_{100}$.

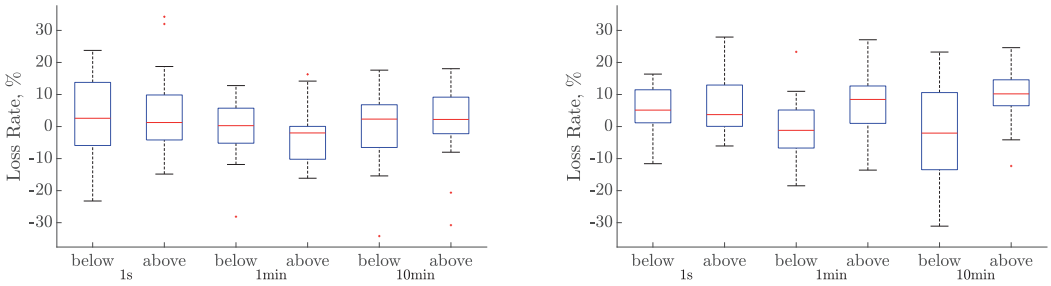
Figure 5.8 – MorphIT$_{100}$ (solid) versus Uniform (dotted) performance given on-off target flows. "Sparse aggregates" scenario: $\phi = 64$ flows/aggregate, $w = 10$sec. The max bin size τ varies.

still introduces significantly more noise (Fig. 5.8b).

5.5.4 The Cost of Differential Privacy

Any mechanism providing differential privacy (like PrivCount) would provide better anonymity than MorphIT; the question we examine is at what cost to report utility.

We simulate the following scenario: An ISP has signed SLAs with 50 customer networks, promising packet loss below 0.1%. During some time interval, each customer network generates a traffic aggregate consisting of $\phi = 512$ real flows. During this time interval, the ISP honors half the SLAs and violates the other half: it introduces packet loss 0.01% to half of these aggregates (we call them "below" aggregates) and packet loss 1% to the other half (we call them "above" aggregates). Each aggregate A crosses the ISP at one ingress and one egress witness; the corresponding traffic reports produced by the two witnesses are used to compute the ISP's packet loss rate with respect to A.



(a) SLA: 0.1%. Real packet loss: 0.01% for "below" aggregates, 1% for "above" aggregates.

(b) SLA: 5%. Real packet loss: 1% for "below" aggregates, 10% for "above" aggregates.

Figure 5.9 – Packet-loss rate estimated from traffic reports anonymized with PrivCount. Time granularity is 1s, 1min, or 10min.

The witnesses use PrivCount to anonymize their traffic reports, with privacy budget $\epsilon = 1, \delta = 10^{-3}$, equally split between the two witnesses, and the sensitivity value (the maximum contribution of a flow to the total aggregate packet count) fixed to 1 packet per msec. We chose these values according to the recommendations in [58].

Fig. 5.9a shows the ISP's packet-loss rates as estimated from the anonymized traffic reports: The first (second) boxplot shows the estimated packet-loss rates for the "below" ("above") aggregates when the reports contain 1s packet counts. The next two boxplots correspond to 1min packet counts, and the last two boxplots to 10min packet counts.

Fig. 5.9b shows similar results for the scenario where the ISP promises packet loss below 5% and introduces packet loss 1% to the "below" aggregates and 10% to the "above" ones.

In both scenarios, the noise needed to guarantee (ϵ, δ)-differential privacy destroys the utility of the traffic reports. Even when the witnesses report a single per-aggregate packet count every 10min (last two boxplots in both figures), the estimated packet-loss rate can differ from the actual one by *tens* of percentage points. With such accuracy levels, it is impossible to determine whether the ISP has honored or violated an SLA for a given aggregate, or whether the ISP is discriminating in favor or against some of the aggregates.

MorphIT also affects report utility but in a controlled manner: A witness that uses MorphIT reports exact per-aggregate counts. Hence, as long as an ISP's entry and exit witness report counts for the same time interval, the ISP's packet-loss rate (and SLA compliance) can be accurately computed during that time interval. Even though witnesses pick their time bins independently from each other, we found that two witnesses that observe the same aggregate pick mostly the same time bins for that aggregate, even in the presence of packet loss. As a result, it is easy to align their traffic reports and compute accurate packet-loss statistics between them.

These results are not surprising, because PrivCount was not designed for anonymizing statistics that are meant to be used for verifying SLA or neutrality compliance. We included them to make

the point that a straightforward application of differential privacy would not work in our context.

5.6 Discussion

We now discuss the limitations of our approach and opportunities for future work.

Privacy guarantees. Our approach does not provide any guarantees about how much a flow can be protected (even if the flow contributes no more than ρ packets during any given time tick). This is because witnesses produce their traffic reports without any coordination, hence there is no guarantee about how much cover their reports provide to each other's flows. Our results indicate that it is possible to protect most flows without coordination. To provide any kind of guarantees, however, we expect that coordination between witnesses is necessary.

Transparency/privacy trade-off. Transparency protocols are useful only if networks cannot arbitrarily exaggerate their performance. Both prior work [18–20, 22, 66, 82] and this thesis (§3 and §4) have shown how the monitor can identify dishonest traffic reports and/or render them ineffective. At the same time, to protect flow privacy, networks need to conceal information from the monitor. Whether this enables networks to be dishonest depends on the nature of the obfuscation. For example, an obfuscation mechanism that naïvely adds high-variance noise to the traffic reports would enable networks to exaggerate their performance. Our obfuscation mechanism is to coarsen the time granularity of the reports. Hence, we do not interfere with the ability of the monitor to identify dishonest reports, but we do force the monitor to detect dishonesty at a coarser time granularity. That said, we think that this trade-off between transparency and privacy can be explored much more deeply and deserves the attention of the research community.

5.7 Related Work

MorphIT shares a common vision with Network Confessional (NC) [20]: give ISPs an interface for supplying quality-of-service feedback to end users. Both proposals are different from approaches to Internet accountability that either resort to alternative Internet protocols, such as AIP [16] and APIP [62], or alternative designs of the whole Internet architecture, such as SCION [70]. However, unlike our protocol, NC does not target anonymity guarantees and may interfere with existing anonymous communication systems, Tor [14, 27], by enabling global passive de-anonymization attacks [26, 48, 60].

Many before us have proposed designs for privacy-preserving data collection in networks [23, 33, 47]. Among them, SEPIA [23] uses secure multi-party computation (SMPC) [77], which allows learning of aggregate network statistics without disclosing local input data, but assumes that learning is secure in itself. Using the optimized SEPIA primitives, one can possibly devise protocols tailored to the applications we consider, i.e., SLA and neutrality verification, however we do not want to restrict a richer class of analysis on the generated traffic reports. Moreover,

applying the SEPIA approach in our context would impose communication overhead and require coordination among networks.

State-of-the-art results in gathering privacy-preserving statistics on anonymity networks were obtained by PrivCount [47]. However, in the context of a transparency protocol, PrivCount is far from providing the report utility we seek (§5.5.4).

To balance anonymity and transparency, we also explored solutions based on "traffic morphing" [80]. By employing convex optimization techniques, witnesses could obfuscate features of traffic reports (e.g., the packet count distribution or the exact pattern of aggregates) while limiting the loss of utility (e.g., minimizing the L1 norm of the difference between the original and noisy aggregate sequences). Nevertheless, we decided not to include it in our final evaluation, as it makes the reports less accurate without the benefit of theoretical privacy guarantees provided by PrivCount.

5.8 Summary

We assess the risk that a basic transparency protocol would pose for low-latency anonymity networks like Tor. We show that there is indeed a risk, in the sense that the traffic reports published by a transparency protocol can help a passive adversary de-anonymize flows. We also show that adding noise to the traffic reports so as to ensure differential privacy would destroy report utility, i.e., make them unusable in the context of a transparency protocol. Instead, we propose MorphIT, an algorithm that coarsens the time granularity of traffic reports in order to obfuscate the flow patterns that are most vulnerable to tracing. We experiment with Poisson and real flows and show that MorphIT significantly improves flow anonymity even in highly adversarial scenarios where there are as few as 64 flows per aggregate.

6 Conclusion

Verifiable network performance leads to informed user decisions and increased network market shares. Existing transparency proposals rely on incentives to remain robust under networks that dishonestly report their performance. However, there is a mismatch between the incentives created by user interests and the incentives required by existing transparency proposals. To solve this mismatch, in this thesis, we propose new incentivization techniques that work for both flexible user interests and for metrics that make performance verification hard. Further, we identify and ease the tussle between networks that seek to prove their performance and anonymity networks that rely on limited performance visibility.

To accurately compute average metrics over user-defined aggregates of packets, we propose a definition of loss and delay means so that a network that lies about a loss/delay mean risks entering conflict with a neighbor (Chapter 3). To accurately estimate jitter despite inaccurate individual performance reports, we identify an interplay between per-aggregate delay means and jitter. Based on this observation, we propose a monitoring technique that groups aggregates based on network policy and overcomes the insufficiency of threat-of-conflict incentives for jitter estimation (Chapter 4). We argue that accurately estimating per-aggregate loss mean, delay mean, and jitter is useful in itself but also a building block for networks to declare meaningful SLAs and for the relevant entities to verify them. Finally, to reconcile the need for both transparency and user anonymity, we propose to adaptively change the granularity of traffic reports to hide sensitive information about individual users, thus improving anonymity (Chapter 5).

We close with a brief discussion on potential research directions.

6.1 Future Work

Loss-based policy verification. An interesting direction is extending our policy verification techniques to loss. We focused here on delay, because prior work did not address it, and we found it to be more challenging to reason about. Future work can adapt our techniques and check whether per-aggregate loss (not just delay) means are exposed to the same process.

E2E measurements or traffic reports? Prior work [28, 30, 37, 38, 53, 56, 64, 75, 83, 84] has built techniques and tools that detected performance issues through end-to-end measurements, potentially crowd-sourced to large user bases through mobile applications. We propose a different approach, where networks report on their own behavior, more reminiscent of prior work on fault localization [18–20, 22, 40, 66, 82]. The ideal solution would probably combine elements of both: Our approach targets more systematic, reliable assessment, but requires the deployment of witnesses at network boundaries. Perhaps by combining it with crowd-sourced E2E measurements we can dramatically decrease the necessary number of witnesses.

Honesty via trusted hardware? We make networks produce correct statements through incentives; an alternative would be to rely (i.e., run the witnesses) on trusted infrastructure. This is an interesting approach, and it faces its own challenges: Existing trusted execution environments, like Intel SGX [13], would not be sufficient—they are designed for commodity hardware, and they provide trusted clocks that work at the granularity of seconds at best [17, 25]. In the future, however, it is plausible to imagine trusted network ASICs with microsecond-accurate trusted clocks. Would it be preferable to rely on a small set of trusted network-hardware manufacturers? or on simpler and less expensive incentives for networks to be honest (as in our approach)? Also, trusted infrastructure residing in untrusted networks can be subject to physical-layer attacks, e.g., tampering with temperature or voltage [61], so incentive-based approaches are still motivated.

Performance-issue localization for application pipelines. Applications become increasingly sophisticated and complex: a single user request arriving at a data center can trigger a chain of requests that hit different machines and invoke a chain of different services. At the same time, keeping the data center highly utilized requires multiplexing different applications over the same resources; but at times of peak demand, this leads to unexpected performance issues. Can we extend the techniques presented in this thesis to localize performance issues to certain machines/services of the chain? This setup brings challenges. Continuously monitoring the performance of each component may be too expensive, and faulty monitoring systems may lead to noisy measurements. We could use ideas from this thesis to extract accurate aggregate metrics from potentially inaccurate, finer-grain, sampled measurements; or localize performance issues to neighboring services of the chain. How do we handle forks (and not only sequences of services, nicely chained one after the other)? Can asynchronous service calls hide performance dependencies on other services? Finally, in a data center, performance matters not only on average but also at the tail; we can get inspiration from our jitter estimation and develop techniques that handle other non-linear metrics like tail latency.

A Proof of Lemma 4.1

Consider a continuous-time stochastic process $U \triangleq \{U(\tau) : \tau \geq O\}$ capturing N_i's (split-responsibility) delay process w.r.t. traffic aggregate A that traverses witness pair j. U takes values in a general space. Consider also a stochastic point process on the interval $[0,\infty)$, characterized by the counting process $N \triangleq \{N(\tau) : \tau \geq 0\}$ or, equivalently, the sequence of successive points $\{T_k : k \geq 1\}$; i.e., $N(\tau) = \sup\{k \geq 0 : T_k \leq \tau\}$, $\tau \geq 0$, where $T_0 = 0$ without there being a 0-th point. Then $U(T_k-) : k \geq 1$ is the sequence of packet delays. We refer to N as the arrival process that samples U at distinct times T_k; i.e, it is our sampling process A^* from §4.2.

We assume that the sample paths of U are left continuous with right limits, while the sample paths of N are right continuous with left limits. As in [21], we also make a mild assumption that is needed for PASTA to hold: the background traffic coinciding with traffic aggregate A is a stationary and ergodic marked point process that is independent of A.

The time average μ of U and its event-average as computed through the packet delays of the arrival process N are:

$$\mu = \frac{1}{\tau}\int_0^\tau U(s)ds,\ \tau > 0, \tag{A.1}$$

$$\widehat{D}_{A^*} = \frac{1}{N(\tau)}\sum_{k=1}^{N(\tau)} U(T_k-),\ N(\tau) > 0 \tag{A.2}$$

According to Theorem 3 in [21] (which is a PASTA statement for the FIFO tandem queueing network model), if N is Poisson, then the two averages are asymptotically equal:

$$\widehat{D}_{A^*} = \mu, \text{ as } \tau \to \infty. \tag{A.3}$$

Eq. (A.3) is similar to a Law of Large Numbers, but a relevant CLT is also applicable when PASTA holds. More specifically, we can use the CLT version for customer and time averages of Proposition 5 from [39], according to which the difference $\tau^{-1/2}\left(\widehat{D}_{A^*} - \mu\right)$ is normally distributed

with zero mean and variance $\lambda^{-1} var(\mathscr{D})$, where λ is the constant stochastic intensity of the Poisson process N, and $\mathscr{D}$ is the limiting sequence of packet delays as time tends to infinity (i.e., $U(T_k) \Rightarrow \mathscr{D}$ as $k \to \infty$)[1].

Given an epoch duration t, the above translates to:

$$\widehat{D}_{A^*} \sim \mathscr{N}\left(\mu, (\lambda t)^{-1} var(\mathscr{D})\right) \tag{A.4}$$

Thus, by taking $\lambda = \frac{n}{t}$ (where $n = |A^*|$ is the number of samples), we conclude the proof.

Note: As mentioned in [21, Section 3], the network setting that is used to prove PASTA for non-intrusive probes (which are similar to our passive-sampling measurements) is far more general than the typically assumed FIFO tandem queueing network. It can include background-traffic streams correlated across witnesses, background-traffic with feedback such as TCP, non-FIFO scheduling disciplines, varying over witnesses, traffic which follows different paths through a network (modeling load balancing), etc., as long as the ergodicity and independence assumptions made above are satisfied and queueing systems act deterministically on the traffic (as happens with FIFO, Weighted Fair Queueing, and processor sharing queueing disciplines).

[1] Also $\mathbb{E}[U(T_k)^2] \to \mathbb{E}[\mathscr{D}^2]$ and $\mathbb{E}[U(T_k)] \to \mathbb{E}[\mathscr{D}] = \mu$ as $k \to \infty$ (see Eq. (27), (28) in [39].

B Aether Details

B.1 Protocol Details

B.1.1 Links and Witnesses

An inter-domain link is the infrastructure between two subsequent (exit and entry) witnesses of neighboring networks, N_i and N_j. In the simplest scenario, N_i and N_j are connected through a direct link; N_i's exit witness is part of the last egress processing point of the last router that a packet encounters in N_i, which operates on the packet right before it is put on the wire; N_j's entry witness is part of the first ingress processing of the first router that a packet encounters in N_j, which operates on the packet right after it is read from the first queue it encounters in the router. In this case, the loss attributed to the inter-domain link is the loss that occurs at N_j's ingress queue, while the delay is the propagation delay between the two witnesses plus the queuing delay at N_j's ingress queue. In more complex scenarios, there is more ambiguity, e.g., the loss and delay attributed to the inter-domain link may have occurred at an ingress queue of the receiving network, or an egress queue of the sending network, or somewhere inside an Internet eXchange point (IXP) connecting the two networks.

We consider this ambiguity both unavoidable and acceptable: Unavoidable, because we cannot realistically impose rules on where exactly—before or after which queue—networks deploy their witnesses. Acceptable, because a transparency protocol does not need to pinpoint the exact queue where packets get lost or delayed; in our proposal, any loss or delay that happens between subsequent witnesses of different networks is *equally* attributed to *both* networks.

Is split responsibility fair? E.g., consider the simplest scenario described above, where N_i and N_j connect through a direct link. Suppose a traffic aggregate experiences significant loss/delay at N_j's ingress queue. Is it fair for N_i to equally share responsibility for it?

We argue that it is: The loss and delay at the ingress queue depends on the queue's size, service rate, and discipline (arguably the responsibility of the receiving network), as well as the nature of the traffic that is sent to the queue (arguably the responsibility of the sending network). When

two networks connect to each other, they agree both on the *properties* of the link between them, and on the nature of the *traffic* that they will exchange through this link. If, at some point, the statements of the two networks indicate that the loss/delay of the inter-domain link is not as expected, this could be either because the receiving network is not managing the link, or because the sending network is not loading the link as agreed (or one of them is lying). What happens next is up to the two networks. In our example, if N_i thinks that the problem lies with N_j (e.g., N_j is not serving the ingress queue at the proper rate, or it has installed a shaper that explicitly delays the given traffic aggregate), then N_i disputes N_j's statements and the two networks enter conflict. This does not mean that N_i is accusing N_j of lying; it means that the loss/delay of the inter-domain link, as indicated by the two networks's statements, is not what N_i expects—which could be due to N_j lying, or to N_j's incorrectly managing the link.

B.1.2 Clock Drift

Witness clock drift has the same effect on the transparency protocol as witness lying. We illustrate with an example: Consider two subsequent witnesses, w_i and w_j, that belong to neighboring networks; suppose w_j's clock lags behind w_i's clock by x microseconds. From w_i's (resp. w_j's) point of view, this scenario is indistinguishable from the one where the two clocks are synchronized and w_j (resp. w_i) is lying to attribute x more microseconds to the inter-domain link between them. If the drift is insignificant relative to the delay normally expected on the link, they may not notice or ignore it; if it is significant, each network will dispute the other's statements (w.r.t. the sampled packets entering/exiting the link), and they will enter conflict.

We do not dictate how networks manage conflict, but we expect that they would start from debugging the relevant inter-domain link and witnesses. In our example, the conflict would be tracked to witness clock drift and resolved with a better clock-synchronization algorithm.

We purposefully did not specify the clock-synchronization algorithm, because it is not a fundamental part of our proposal. Networks have incentives to keep witnesses synchronized: such that the monitor correctly computes their performance, and to avoid unnecessary conflict with neighbors. Each network can choose the resources it puts into clock synchronization as a function of how much it cares for these incentives.

B.1.3 Estimation and Accuracy

Loss-mean confidence interval. Computing a confidence interval for the loss-mean estimate requires a loss model. The simplest one is Bernoulli loss, which assumes that A's packets are dropped independently from each other and with probability $\widehat{LR}_i(A)$. Despite its simplicity (and despite the fact that actual packet losses are not independent), this model works well enough in many practical scenarios, and we found that to be the case in our experiments as well. So: Given a Bernoulli loss model for A, the monitor computes a classic Gaussian confidence interval at a

level γ (say 95%) as [51, Thm 2.2]:

$$\widehat{LR}_i(A) \pm \eta_\gamma \sqrt{\frac{\widehat{LR}_i(A)\,(1-\widehat{LR}_i(A))}{k}},$$

where η_γ is the $\frac{1+\gamma}{2}$-quantile of the standard normal distribution, and k is the (split-responsibility) number of packets that entered network N_i.[1] However, the monitor could plug in any loss model. E.g., a more sophisticated option would be Gilbert loss, which models loss as a Markov process that transitions between a "loss-less" and a "lossy" state [46].

Delay-mean confidence interval. In principle, the monitor can compute a similar Gaussian confidence interval as above in the following way: Let $\mathscr{D}$ be the empirical (split-responsibility) delay distribution experienced by A in N_i. Let σ^2 be $\mathscr{D}$'s variance. If n samples are drawn i.i.d. from $\mathscr{D}$, a confidence interval for the delay-mean estimate at a level γ is:

$$\widehat{D}_i(A) \pm \eta_\gamma \frac{\hat{\sigma}}{\sqrt{n}},$$

where η_γ is defined as above, and $\hat{\sigma}$ is the sample standard deviation. The latter, however, is not a trusted estimate for σ, because it is obtained from the (unreliable) statements.

B.2 Evaluation Details

B.2.1 Monitor Resources

Statement mapping and normalization. Using one core, our implementation takes 3.60µsec, on average, to ingest a witness statement and map it to a traffic aggregate, and to normalize the resulting aggregates. Let's assume that the target ISP has 6 neighbors [69], hence 6 witnesses and 30 witness pairs (counting each witness pair twice, once per traffic direction). Let's assume that each defined traffic aggregate traverses, on average, 6 networks, i.e., 10 witnesses in total (average AS-path length on the Internet is below 4, and we add two edge networks assuming that they are not their own ASes). This means that, during each epoch, the monitor collects statements from 300 witnesses. If we assume an average packet size of 891 bytes (as in the CAIDA traces) and a sampling rate of 5%, the monitor receives ~2M statements per second for every Gbps of observed traffic. Hence, we need 10 cores per Gbps of observed traffic to do the statement mapping and aggregate normalization in real time.

Policy verification. Using one core, our implementation takes 1.31sec, on average, to validate the traffic policy of one witness pair that implements one traffic class. So, if the target ISP has 30 witness pairs, each implementing one traffic class, we need 40 cores to validate its policy in real time.

[1] k can be computed based on m, i.e., the number of A's sampled packets that exited the source network, and the per-aggregate loss means.

B.2.2 Accuracy: Sensitivity Analysis

In our evaluation (§4.4), we presented experiments where the monitor relies on 128 traffic aggregates and 64 packets per aggregate to estimate network N_i's performance; we now justify the choice of these two parameters.

Figs. 4.2c and 4.2d show the monitor's delay-mean error and jitter error, for different aggregate numbers and sizes. In these experiments, congestion factor = 1, and epoch length = 256sec. As expected, increasing the number of aggregate improves the jitter estimate, however, the improvement is not significant beyond 128 aggregates. On the other hand, increasing the number of packets per aggregate (beyond 64) does not improve the estimates. (Though we should note that decreasing it below 32 does worsen the estimates.)

Figs. 4.3a and 4.3b show how often the monitor succeeds in computing a jitter estimate, for different aggregate numbers and sizes. In these experiments, congestion factor = 1, and epoch length = 256sec. We see that using more aggregates actually *reduces* the probability of success. Part of this is intuitive: the more aggregates the monitor requires to compute an estimate, the less likely it is that it will find them in the target epoch (Fig. 4.3a). But there is also a counter-intuitive part: when the monitor uses more than a few tens of aggregates to run the normality test, the accuracy of the test *decreases* with the number of aggregates (Fig. 4.3b). This is due to a known artifact of the Shapiro-Wilk normality test, which suffers an unexpected false-positive rate when the number of random sets increases beyond a few tens. Bottom line: If the monitor wants to leverage more aggregates, it is better to use a CLT-based normality test other than Shapiro-Wilk, e.g., ANOVA [36].

C MorphIT Processing Overhead

The online version of our algorithm operates on active windows of size ω, and its runtime should scale linearly with the number of active windows within the observation window.

Fig. C.1 shows the average runtime of MorphIT$_{100}$ as a function of the maximum bin size τ. There are $\phi = 512$ flows per aggregate, and the adversary's observation window size is $w = 10$s. The implementation is in Matlab, and it is running on an Intel Core i7-7700 3.6GHz CPU.

The figure confirms our algorithm's scalability. Moreover, given that our implementation is far from optimized for performance, and it is running on a single core, these numbers indicate that our algorithm is deployable.

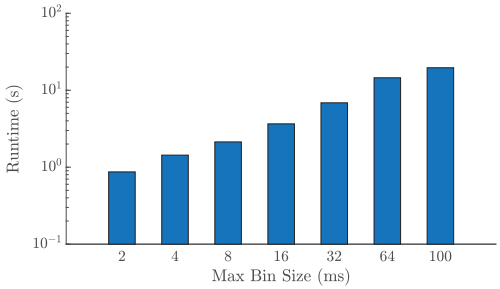


Figure C.1 – Average runtime of MorphIT$_{100}$. $\phi = 512$ flows per aggregate. $w = 10$s.

Bibliography

[1] AT&T Service Level Agreement (SLA). http://cpr.att.com/pdf/se/0001-0003.pdf. accessed Sep 2021.

[2] BEREC. https://www.berec.europa.eu/. accessed Jan 2022.

[3] The CAIDA UCSD Anonymized Internet Traces - 20140320 & 20140619 & 20150219 & 20150521 & 20150917 & 20151217 & 20160121 & 20160218 & 20160317 & 20160406 & 20180315 & 20180419 & 20180517 & 20180621 & 20180719 & 20180816 & 20180921 & 20181018 & 20181115 & 20181220 & 20190117. https://www.caida.org/catalog/datasets/passive_dataset.

[4] During Netflix money fight, Cogent's other big customers suffered too. https://arstechnica.com/information-technology/2014/11/during-netflix-money-fight-cogents-other-big-customers-suffered-too/. accessed Sep 2021.

[5] Comcast SLA for Wholesale Dedicated Internet. https://www.comcasttechnologysolutions.com/sites/default/files/2016-09/Service%20Level%20Agreement.pdf, . accessed Sep 2021.

[6] Netflix's Disputes With Verizon, Comcast Under Investigation. https://time.com/2871498/fcc-investigates-netflix-verizon-comcast/, . accessed Sep 2021.

[7] Comcast gets paid by Netflix and might still want money from Cogent. https://arstechnica.com/information-technology/2014/02/comcast-gets-paid-by-netflix-and-might-still-want-money-from-cogent/. accessed Sep 2021.

[8] Open Internet Regulation. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32015R2120&rid=2. accessed June 2021.

[9] FCC. https://www.fcc.gov/, . accessed Jan 2022.

[10] Protecting and Promoting the Open Internet. https://www.federalregister.gov/documents/2015/04/13/2015-07841/protecting-and-promoting-the-open-internet, . accessed June 2021.

[11] Restoring Internet Freedom Order. https://www.fcc.gov/document/fcc-releases-restoring-internet-freedom-order, . accessed Sep 2021.

[12] ICANN. https://www.icann.org/. accessed Jan 2022.

[13] Intel Software Guard Extensions. https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html. accessed June 2021.

[14] The Tor Project. https://www.torproject.org/. accessed May 2022.

[15] Trump's FCC has revealed plans to wipe out net neutrality. https://www.vox.com/2017/11/21/16679114/fcc-ajit-pai-net-neutrality-rules-donald-trump. accessed Sep 2021.

[16] David G. Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. Accountable Internet Protocol (Aip). In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, page 339–350, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581750. doi: 10.1145/1402958.1402997. URL https://doi.org/10.1145/1402958.1402997.

[17] Fatima M. Anwar and Mani B. Srivastava. Applications and Challenges in Securing Time. 2019. https://www.usenix.org/conference/cset19/presentation/anwar.

[18] Katerina Argyraki, Petros Maniatis, David Cheriton, and Scott Shenker. Providing packet obituaries. In *Proceedings of the ACM Workshop on Hot Topics in Networking (HotNets)*, 2004.

[19] Katerina Argyraki, Petros Maniatis, Olga Irzak, Subramanian Ashish, and Scott Shenker. Loss and Delay Accountability for the Internet. In *2007 IEEE International Conference on Network Protocols*, pages 194–205, 2007. doi: 10.1109/ICNP.2007.4375850.

[20] Katerina Argyraki, Petros Maniatis, and Ankit Singla. Verifiable network-performance measurements. In *Proceedings of the 6th International COnference*, pages 1–12, 2010.

[21] Francçois Baccelli, Sridhar Machiraju, Darryl Veitch, and Jean C. Bolot. The Role of PASTA in Network Measurement. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, page 231–242, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933085. doi: 10.1145/1159913.1159940. URL https://doi.org/10.1145/1159913.1159940.

[22] Boaz Barak, Sharon Goldberg, and David Xiao. Protocols and Lower Bounds for Failure Localization in the Internet. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 341–360. Springer, 2008. doi: 10.1007/978-3-540-78967-3_20. URL https://doi.org/10.1007/978-3-540-78967-3_20.

[23] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, page 15, USA, 2010. USENIX Association. ISBN 8887666655554.

[24] Arturo Carrillo. Are There Universal Standards for Network Neutrality? *University of Pittsburgh Law Review*, 80(4), 2019. ISSN 1942-8405. doi: 10.5195/lawreview.2019.654. https://lawreview.law.pitt.edu/ojs/index.php/lawreview/article/view/654.

[25] Shanwei Cen and Bo Zhang. Trusted Time and Monotonic Counters with Intel® Software Guard Extensions Platform Services. In *Intel Resource Library*, 2017. https://community.intel.com/legacyfs/online/drupal_files/managed/1b/a2/Intel-SGX-Platform-Services.pdf.

[26] George Danezis. The Traffic Analysis of Continuous-Time Mixes. In *Proceedings of the 4th International Conference on Privacy Enhancing Technologies*, PET'04, page 35–50, Berlin, Heidelberg, 2004. Springer-Verlag. ISBN 3540262032. doi: 10.1007/11423409_3. URL https://doi.org/10.1007/11423409_3.

[27] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium (USENIX Security 04)*, San Diego, CA, August 2004. USENIX Association. URL https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

[28] Marcel Dischinger, Alan Mislove, Andreas Haeberlen, and Krishna P. Gummadi. Detecting Bittorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, IMC '08, page 3–8, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605583341. doi: 10.1145/1452520.1452523. URL https://doi.org/10.1145/1452520.1452523.

[29] N. G. Duffield and Matthias Grossglauser. Trajectory Sampling for Direct Traffic Observation. *IEEE/ACM Trans. Netw.*, 9(3):280–292, June 2001. ISSN 1063-6692. doi: 10.1109/90.929851. URL https://doi.org/10.1109/90.929851.

[30] Nick Duffield. Network Tomography of Binary Network Performance Characteristics. *IEEE Transactions on Information Theory*, 52(12):5373–5388, 2006. doi: 10.1109/TIT.2006.885460.

[31] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 486–503. Springer, 2006.

[32] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[33] Tariq Elahi, George Danezis, and Ian Goldberg. PrivEx: Private Collection of Traffic Statistics for Anonymous Communication Networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 1068–1079, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329576. doi: 10.1145/2660267.2660280. URL https://doi.org/10.1145/2660267.2660280.

[34] Nathan S. Evans, Roger Dingledine, and Christian Grothoff. A Practical Congestion Attack on Tor Using Long Paths. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, page 33–50, USA, 2009. USENIX Association.

[35] Nick Feamster and Hari Balakrishnan. Packet loss recovery for streaming video.

[36] Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in statistics*, pages 66–70. Springer, 1992.

[37] Denisa Ghita, Katerina Argyraki, and Patrick Thiran. Network Tomography on Correlated Links. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, page 225–238, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450304832. doi: 10.1145/1879141.1879170. URL https://doi.org/10.1145/1879141.1879170.

[38] Denisa Ghita, Can Karakus, Katerina Argyraki, and Patrick Thiran. Shifting Network Tomography toward a Practical Goal. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450310413. doi: 10.1145/2079296.2079320. URL https://doi.org/10.1145/2079296.2079320.

[39] Peter W. Glynn, Benjamin Melamed, and Ward Whitt. Estimating Customer and Time Averages. *Operations Research*, 41(2):400–408, 1993. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/171786.

[40] Sharon Goldberg, David Xiao, Eran Tromer, Boaz Barak, and Jennifer Rexford. Path-Quality Monitoring in the Presence of Adversaries. In *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '08, page 193–204, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580050. doi: 10.1145/1375457.1375480. URL https://doi.org/10.1145/1375457.1375480.

[41] David Goulet, Aaron Johnson, George Kadianakis, and Karsten Loesing. Hidden-service statistics reported by relays. Technical report, NAVAL RESEARCH LAB WASHINGTON DC, 2015.

[42] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[43] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[44] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. PeerReview: Practical Accountability for Distributed Systems. In *Proceedings of Twenty-First ACM SIGOPS Symposium on*

Operating Systems Principles, SOSP '07, page 175–188, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595935915. doi: 10.1145/1294261.1294279. URL https://doi.org/10.1145/1294261.1294279.

[45] Andreas Haeberlen, Ioannis Avramopoulos, Jennifer Rexford, and Peter Druschel. NetReview: Detecting When Interdomain Routing Goes Wrong. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, page 437–452, USA, 2009. USENIX Association.

[46] Gerhard Hasslinger and Oliver Hohlfeld. The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet. In *Proc. of the GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems*, MMB, March 2008.

[47] Rob Jansen and Aaron Johnson. Safely Measuring Tor. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1553–1567, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978310. URL https://doi.org/10.1145/2976749.2978310.

[48] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 337–348, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324779. doi: 10.1145/2508859.2516651. URL https://doi.org/10.1145/2508859.2516651.

[49] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A nonstationary Poisson view of Internet traffic. In *IEEE INFOCOM 2004*, volume 3, pages 1558–1569 vol.3, 2004. doi: 10.1109/INFCOM.2004.1354569.

[50] Stevens Le Blond, David Choffnes, Wenxuan Zhou, Peter Druschel, Hitesh Ballani, and Paul Francis. Towards Efficient Traffic-Analysis Resistant Anonymity Networks. *SIGCOMM Comput. Commun. Rev.*, 43(4):303–314, aug 2013. ISSN 0146-4833. doi: 10.1145/2534169.2486002. URL https://doi.org/10.1145/2534169.2486002.

[51] Jean-Yves Le Boudec. *Performance Evaluation of Computer and Communication Systems*. EPFL Press, Lausanne, 2010. URL http://infoscience.epfl.ch/record/146812.

[52] Christoph Lenzen, Philipp Sommer, and Roger Wattenhofer. Optimal Clock Synchronization in Networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, page 225–238, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585192. doi: 10.1145/1644038.1644061. URL https://doi.org/10.1145/1644038.1644061.

[53] Fangfan Li, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. A Large-Scale Analysis of Deployed Traffic Differentiation Practices. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 130–144, New

York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359566. doi: 10.1145/3341302.3342092. URL https://doi.org/10.1145/3341302.3342092.

[54] Linux. ping(8) — Linux manual page. https://man7.org/linux/man-pages/man8/ping.8.html, . accessed May 2022.

[55] Linux. traceroute(8) — Linux manual page. https://man7.org/linux/man-pages/man8/traceroute.8.html, . accessed May 2022.

[56] Ovidiu Sebastian Mara. Network Neutrality Inference using Network Tomography. page 125, 2018. doi: 10.5075/epfl-thesis-8076. URL http://infoscience.epfl.ch/record/253617.

[57] Apostolaki Maria, Zohar Aviv, and Vanbever Laurent. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017.

[58] Frank D. McSherry. Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, page 19–30, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585512. doi: 10.1145/1559845.1559850. URL https://doi.org/10.1145/1559845.1559850.

[59] Microsoft. Media Quality and Network Connectivity Performance in Microsoft Teams. https://docs.microsoft.com/en-us/skypeforbusiness/optimizing-your-network/media-quality-and-network-connectivity-performance. accessed May 2022.

[60] Steven J. Murdoch and Piotr Zieliński. Sampled Traffic Analysis by Internet-Exchange-Level Adversaries. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, PET'07, page 167–183, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3540755500.

[61] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. Plundervolt: Software-based Fault Injection Attacks against Intel SGX. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1466–1482, 2020. doi: 10.1109/SP40000.2020.00057.

[62] David Naylor, Matthew K. Mukerjee, and Peter Steenkiste. Balancing accountability and privacy in the network. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, page 75–86, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328364. doi: 10.1145/2619239.2626306. URL https://doi.org/10.1145/2619239.2626306.

[63] Netflix. A cooperative approach to content delivery. https://openconnect.netflix.com/Open-Connect-Briefing-Paper.pdf. accessed May 2022.

[64] H. X. Nguyen and P. Thiran. The Boolean Solution to the Congested IP Link Location Problem: Theory and Practice. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 2117–2125, 2007. doi: 10.1109/INFCOM.2007.245.

[65] Pavlos Nikolopoulos. Traffic Receipts for Network Transparency. page 114, 2018. doi: 10.5075/epfl-thesis-8904. URL http://infoscience.epfl.ch/record/261217.

[66] Pavlos Nikolopoulos, Christos Pappas, Katerina Argyraki, and Adrian Perrig. Retroactive Packet Sampling for Traffic Receipts. *SIGMETRICS Perform. Eval. Rev.*, 47(1):17–18, December 2019. ISSN 0163-5999. doi: 10.1145/3376930.3376942. URL https://doi.org/10.1145/3376930.3376942.

[67] Christos Pappas, Katerina Argyraki, Stefan Bechtold, and Adrian Perrig. Transparency Instead of Neutrality. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, HotNets-XIV, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340472. doi: 10.1145/2834050.2834082. URL https://doi.org/10.1145/2834050.2834082.

[68] Vern Paxson. End-to-End Routing Behavior in the Internet. SIGCOMM '96, page 25–38, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917901. doi: 10.1145/248156.248160. URL https://doi.org/10.1145/248156.248160.

[69] Aleksi Peltonen. A Map of The Internet. Technical report, 2017. https://csperkins.org/research/routing/2017-05-01-peltonen-project/report.pdf.

[70] Adrian Perrig, Pawel Szalachowski, Raphael M. Reischuk, and Laurent Chuat. *SCION: A Secure Internet Architecture*. Springer International Publishing AG, 2017. ISBN 978-3-319-67079-9. doi: 10.1007/978-3-319-67080-5. URL /publications/papers/SCION-book.pdf.

[71] Andreas Pfitzmann and Marit Köhntopp. Anonymity, Unobservability, and Pseudeonymity — a Proposal for Terminology. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, page 1–9, Berlin, Heidelberg, 2001. Springer-Verlag. ISBN 3540417249.

[72] Matthew Prince. August 30th 2020: Analysis of CenturyLink/Level(3) Outage. https://blog.cloudflare.com/analysis-of-todays-centurylink-level-3-outage/. accessed May 2022.

[73] Vibhor Rastogi and Suman Nath. Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, page 735–746, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300322. doi: 10.1145/1807167.1807247. URL https://doi.org/10.1145/1807167.1807247.

[74] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security*, pages 18–33. Springer, 2006.

[75] Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, and Mostafa Ammar. Detecting Network Neutrality Violations with Causal Inference. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*,

CoNEXT '09, page 289–300, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605586366. doi: 10.1145/1658939.1658972. URL https://doi.org/10.1145/1658939.1658972.

[76] Steven J. Vaughan-Nichols. Why Atlassian Failed So Hard. https://thenewstack.io/why-atlassian-failed-so-hard/. accessed May 2022.

[77] Wikipedia. Secure multi-party computation. https://en.wikipedia.org/wiki/Secure_multi-party_computation, . accessed May 2022.

[78] Wikipedia. Packet Loss. https://en.wikipedia.org/wiki/Packet_loss, . accessed May 2022.

[79] Ronald W. Wolff. Poisson Arrivals See Time Averages. *Operations Research*, 30(2): 223–231, 1982. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/170165.

[80] Charles V Wright, Scott E Coull, and Fabian Monrose. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In *NDSS*, volume 9. Citeseer, 2009.

[81] Xin Zhang, Abhishek Jain, and Adrian Perrig. Packet-Dropping Adversary Identification for Data Plane Security. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582108. doi: 10.1145/1544012.1544036. URL https://doi.org/10.1145/1544012.1544036.

[82] Xin Zhang, Zongwei Zhou, Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Adrian Perrig, and Patrick Tague. ShortMAC: Efficient Data-Plane Fault Localization. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012. URL https://www.ndss-symposium.org/ndss2012/shortmac-efficient-data-plane-fault-localization.

[83] Ying Zhang, Z. Morley Mao, and Ming Zhang. Ascertaining the Reality of Network Neutrality Violation in Backbone ISPs. In *HotNets*. Association for Computing Machinery, Inc., January 2008. URL https://www.microsoft.com/en-us/research/publication/ascertaining-the-reality-of-network-neutrality-violation-in-backbone-isps/.

[84] Zhiyong Zhang, Ovidiu Mara, and Katerina Argyraki. Network Neutrality Inference. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, page 63–74, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328364. doi: 10.1145/2619239.2626308. URL https://doi.org/10.1145/2619239.2626308.

[85] Wenchao Zhou, Qiong Fei, Arjun Narayan, Andreas Haeberlen, Boon Thau Loo, and Micah Sherr. Secure Network Provenance. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, page 295–310, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450309776. doi: 10.1145/2043556.2043584. URL https://doi.org/10.1145/2043556.2043584.

[86] Wenchao Zhou, Suyog Mapara, Yiqing Ren, Yang Li, Andreas Haeberlen, Zachary Ives, Boon Thau Loo, and Micah Sherr. Distributed Time-Aware Provenance. *Proc. VLDB*

Endow., 6(2):49–60, December 2012. ISSN 2150-8097. doi: 10.14778/2535568.2448939. URL https://doi.org/10.14778/2535568.2448939.

Georgia Fragkouli

Curriculum Vitae

+41 78 676 12 81
georgia.fragkouli@epfl.ch

Research Interests

Networked systems, in particular their security, privacy, and performance monitoring

Education

2016–2022 **Ph.D. in Computer Science**
École Polytechnique Fédérale de Lausanne (EPFL)
Thesis: Toward Internet Performance Transparency
Advisors: Katerina Argyraki and Bryan Ford

2010–2016 **MEng Electrical and Computer Engineering**
National Technical University of Athens (NTUA)
Thesis: Performance Modeling of Multi-Channel Networks with Multiple Receivers per Node
Advisor: Efstathios Sykas

Awards

2021 **EPFL IC Teaching Assistant Award**

2020 **EPFL IC Academic Excellence Award**

2020 **IETF/IRTF Applied Networking Research Prize (ANRP)** for our work on reconciling anonymity with Internet performance transparency

2016–2017 **EPFL IC Ph.D. Fellowship**

2010 **Award of Excellence** by Eurobank for top marks in the university entrance exams

2009 **Award of Excellence** by the Greek Ministry of Education for the best score across high schools in the Chios district

Publications

Under Preparation Internet Performance Transparency
Georgia Fragkouli, Pavlos Nikolopoulos, and Katerina Argyraki

Under Preparation Limiting Lamport Exposure to Distant Failures in Globally-Managed Distributed Systems
Cristina Basescu, Georgia Fragkouli, Enis Ceyhun Alp, Vero Estrada-Galiñanes, and Bryan Ford

PETS 2019 MorphIT: Morphing Packet Reports for Internet Transparency
Georgia Fragkouli, Katerina Argyraki, and Bryan Ford, **IETF/IRTF ANRP Award**

HotOS 2019 Rethinking General-Purpose Decentralized Computing
Enis Ceyhun Alp, Eleftherios Kokoris-Kogias, Georgia Fragkouli, and Bryan Ford

Invited Presentations

2021 & 2022 EPFL IC Open House, Lightning talk

2022 Google's Networking Research Summit, Lightning talk

2020 IETF 109 Meeting, Award talk

2019 Microsoft's Research Workshop on Next-Generation Cloud Infrastructure, Poster

2019 PETS, Conference talk

2019 EcoCloud Annual Event, Research talk

Teaching Assistantships

Fall 2017–2021 Computer Networks (EPFL, COM-208)
Spring 2020–2021 Introduction to Database Systems (EPFL, CS-322)
Fall 2011 Introduction to Computer Programming (NTUA, 3.4.01.1)

Programming Languages

C++, MATLAB, Python

Languages

English, Greek (Native)

Extracurricular

2022 **Volunteer** at the EPFL Applied Machine Learning Days (AMLD)
2015–2016 **Fundraising Volunteer** at Job Fair Athens
2009 **Literature Award** by Faros Varvasiou for 2nd place at the 7th Literature and Poetry Student Competition about the Greco-Turkish War
2007 **Piano Teaching Diploma** (Ptychio)

Ce document a été imprimé au Centre d’impression EPFL,
imprimerie climatiquement neutre, certifiée myClimate.