Generating Realistic Application Workloads for Mix-Based Systems for Controllable, Repeatable and Usable Experimentation

Karl-Peter Fuchs, Dominik Herrmann, Hannes Federrath

Generating Realistic Application Workloads for Mix-Based Systems for Controllable, Repeatable and Usable Experimentation

Karl-Peter Fuchs, Dominik Herrmann, and Hannes Federrath

University of Hamburg, Computer Science Department, Germany

Abstract. Evaluating and improving the performance of anonymity systems in a real-world setting is critical to foster their adoption. However, current research in this field mostly employs unrealistic models for evaluation purposes. Moreover, previously documented results are often difficult to reproduce. We propose two complementary workload models that operate on network traces in order to improve the evaluation of anonymity systems. In comparison to other approaches our workload models are more realistic, as they derive characteristics from trace files recorded in real networks and preserve dependencies of the flows of individual hosts. We also describe our ready-to-use open source evaluation suite that implements our models. Given our tools, researchers can easily create and re-use well-defined workload sets for evaluation purposes. Finally, we demonstrate the importance of realistic workload models by evaluating a well-known dummy traffic scheme with our tools.

1 Introduction

Mix-based anonymity systems have become an important technology to protect the privacy of users on the Internet. Since the original proposal by David Chaum in 1981 [9] a large number of mixing schemes for various application areas has been published. Especially low-latency anonymity services like Tor [11] and JAP (JonDonym) [7] have found widespread adoption.

The security and performance evaluation of such systems is challenging because of their complex construction and dynamic nature: Typically they consist of multiple nodes distributed on the Internet, which interact with each other, with a set of clients and (usually) a set of servers. Analytically derived statements obtained by mathematical proofs or queuing theory serve as an important foundation in this field. However, analytical results cannot reliably predict the behaviour and performance of a system once real users adopt it in practice. Simulations with realistic traffic are essential to obtain significant results.

Nevertheless, we observe that some researchers in the privacy-enhancing technologies (PET) community struggle with the evaluation of their proposals: On the one hand, some publications lack an evaluation in a practical setting, and, on the other hand, practically deployed systems such as Tor are sometimes evaluated with quite unrealistic traffic models. Moreover, different datasets are used

for evaluation and some papers lack important details regarding the employed preprocessing or sampling technique.

We believe these deficiencies are mainly due to the lack of *appropriate standard workload models* and the fact that there is no *easily accessible, well-established evaluation procedure* in the PET research area. As a consequence there is a huge gap between theory and practice and published results are difficult to compare to each other. The **contribution of this paper** is threefold: **Firstly**, we propose a dependency-preserving model for workload extraction from Internet trace files that is suitable for the evaluation of low-latency anonymity systems (DPE Model). **Secondly**, we propose a replay and feedback model for traffic generation that takes into account the latencies of the evaluated system (R&F Model). **Thirdly**, we describe our workload generation tool that allows researchers to create or reproduce well-defined evaluation scenarios (Reproducible Scenario Builder). We have integrated these three components into an evaluation suite that has been released as open source software under the GPLv3 at `https://www.informatik.uni-hamburg.de/SVS/gmix/`.

The rest of this paper is structured as follows: In Sect. 2 we review related work before we outline our design goals and the construction of our workload model in Sect. 3. In Sect. 4 we describe our evaluation suite, which includes implementations of the DPE and R&F models as well as the Reproducible Scenario Builder. Finally, in Sect. 5 we present results from empirical evaluations that indicate that our models generate realistic traffic. We also demonstrate the importance of realistic workload models, before we conclude in Sect. 6.

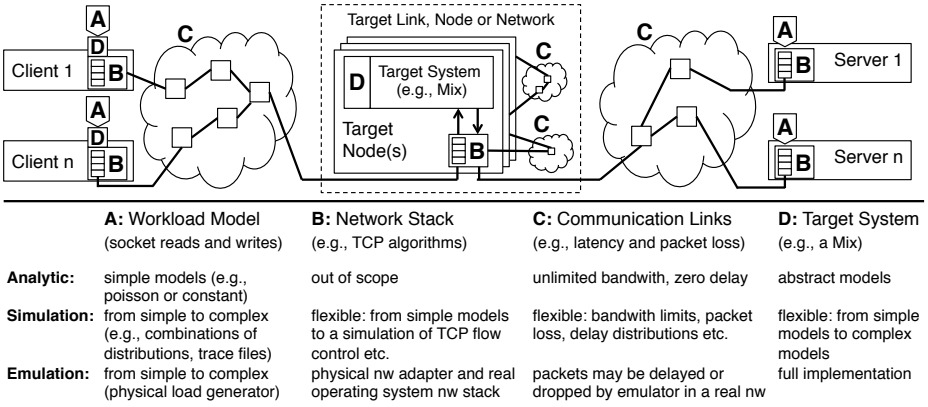
2 Fundamentals and Related Work

Our contribution, a trace-driven workload model for the evaluation of anonymity systems, relates to two fields, *network research* and *privacy-enhancing technologies*. In this section we review the most relevant efforts from these two areas. We also identify shortcomings of the existing approaches that motivate our work.

2.1 Evaluation of Distributed Systems

Figure 1 sketches the components needed for the evaluation of a distributed system. The evaluation can be performed with different levels of abstraction: (1) studying a proposed system *analytically* (e. g., via mathematical proofs or queuing theory), (2) modelling (parts of) the proposed system and its environment and validating the analytical results within *simulations*, and (3) implementing the proposal and measuring its performance in an *emulated network* or a *real-world setting*. In each case *models* can be used to control certain aspects of the proposed system or certain influence factors of the environment.

The *network research community* has brought up several mature and approved models and implementations, e. g., the network simulators ns-2, ns-3, SSF, OPNET and OMNeT++ (providing models for Components A, B and C in Fig. 1), the virtual network emulators Modelnet and Emulab (Components B and C) or the workload generation tools Tmix and Swing (Component A).



	A: Workload Model (socket reads and writes)	**B:** Network Stack (e.g., TCP algorithms)	**C:** Communication Links (e.g., latency and packet loss)	**D:** Target System (e.g., a Mix)
Analytic:	simple models (e.g., poisson or constant)	out of scope	unlimited bandwith, zero delay	abstract models
Simulation:	from simple to complex (e.g., combinations of distributions, trace files)	flexible: from simple models to a simulation of TCP flow control etc.	flexible: bandwith limits, packet loss, delay distributions etc.	flexible: from simple models to complex models
Emulation:	from simple to complex (physical load generator)	physical nw adapter and real operating system nw stack	packets may be delayed or dropped by emulator in a real nw	full implementation

Fig. 1. Models typically involved in evaluation of distributed systems

2.2 Existing Approaches for the Evaluation of PETs

The *PET community* has started to adapt and extend these solutions for the evaluation of anonymity systems. Noticeable examples are the network simulator *Shadow* [17] and the emulation testbed *ExperimenTor* [5]. Both systems try to accurately model the topology and routing mechanism (Component C) of the Tor network [11], which is the most popular anonymity system at the moment. Shadow employs realistic models for the network stack (Component B), and ExperimenTor even uses physical hardware for this part. Both approaches make use of the actual Tor implementation for experimentation (Component D). In [14] we have introduced the *gMix framework* that focuses on the implementation of Component D, i. e., it facilitates building customized anonymity systems from ready-to-use implementations (plug-ins) of previously suggested mix concepts. Like ExperimenTor, gMix can be used in conjunction with a virtual network emulator. Additionally, it provides a basic discrete-event network simulator for abstract but fast evaluations (Components B and C).

Workload Models The approaches mentioned in the previous paragraph make use of quite sophisticated models for Components B, C and D. However, they employ only very basic workload models (Component A): The gMix framework only supports basic statistical distributions, and recent studies using Shadow and ExperimenTor rely on a simple *on–off* workload model: In *on* phases, clients retrieve files of different size in varying intervals. File sizes are chosen to match typical web page sizes [20, 23]. Intervals are drawn from a distribution obtained in a 2003 study [15, 17] or at random with an upper bound of 11 seconds [23]. Others (cf. [25]) simply *pick up* HTTP flows from a trace file and replay them successively for each client (simplex, *open loop* [13]).

These workload models are a strong simplification of the actual events taking place when a user browses through the WWW, which is one of the most popular

applications anonymity systems are used for [18, 27]. Actually, downloading a typical web page requires the web browser to handle multiple request–response pairs and parallel connections (a more detailed description follows in Sect. 3.4). In contrast to real-world implementations, the simplistic workload models used in these studies assume that web pages are retrieved within a single roundtrip or within a single TCP connection. Note that this discrepancy does not necessarily mean that the results obtained in [17, 23, 25] are wrong (the authors do consider the limitations of their models when drawing conclusions). As we strive for more realistic evaluations and we want to validate and compare the results obtained in previous studies, we have designed a more comprehensive and accurate traffic model which will be described in the next section.

3 Designing a Workload Model for Anonymity Systems

Performance evaluations consist of observing the system under test while it handles a specific workload. In their seminal paper Agrawala et al. [1] describe the application of workload models for the evaluation of the performance of computers. Instead of live workloads, workload models are used to generate synthetic workloads that can be replayed multiple times. A realistic **workload model** is supposed to capture both the behaviour of the users that are issuing requests to a system as well as the load these requests induce on the system under test. Today, workload models play an important role to analyse distributed systems. Creating realistic network traffic for experimentation is a well-studied subject in the network research community. However, the applicability of these models and tools for the context of anonymity systems is diverging.

Our workload model consists of two complementary parts: the "Dependency-Preserving Extraction (DPE) Model" and the "Replay and Feedback (R&F) Model". The DPE Model is used to extract flows in a dependency-preserving manner from trace files. Moreover, it captures behavioral characteristics of the individual hosts. The R&F Model determines how traffic is replayed during evaluation, taking into account feedback from the system under test.

In Sect. 3.1 we review the structure of the system under test we are interested in, namely low-latency anonymity systems. In Sects. 3.2 and 3.3 we outline the overall goals that motivated our design decisions for our workload model. After that we will describe our two complementary workload models, the DPE Model (Sect. 3.4) and the R&F model (Sect. 3.5).

3.1 Characteristics of Anonymity Systems

Figure 2 shows the typical architecture of an anonymity service (cf. [7, 11, 14]). Mixes and clients form an overlay network. Connections of user applications (e. g., web browsers) are multiplexed and routed via several mix nodes before they are forwarded to their destinations (e. g., to a web server). Clients apply a layer of encryption for each mix to assert bitwise unlinkability. Mix servers are distributed across the Internet and communicate via TCP or UDP. Mixes

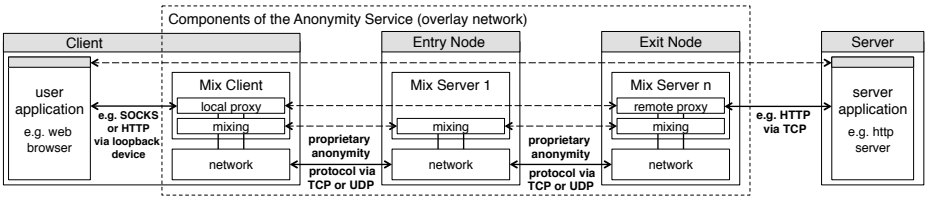


Fig. 2. Typical architecture of an anonymity service

delay messages to build an anonymity set (*output strategy*, cf. [9, 14]). Congestion causes further delays in deployed anonymity systems (cf. [10]). The typical delay is on the order of a few seconds (cf. [10, 27]). Given a certain level of privacy, maximizing throughput and minimizing user-perceived latency are the primary objectives during the design of anonymity systems.

Workload modelling for anonymity systems differs fundamentally from the objectives typically encountered in network research, where the goal is often to create realistic workloads for a single server or a realistic (background) traffic mix for a single target link (so-called *dumbbell topology*) on the packet level (cf., for instance, [22]). As a result, most tools from the network community cannot be used for the evaluation of PETs without modification. However, the traffic modelling approaches codified in those tools may still be applicable, though.

3.2 Design Goals

Our contribution has been guided by the following goals. Our main objective is to provide a more **realistic** workload model (in comparison to the models used by the PET community at the moment, cf. Sect. 2.2). Researchers should be able to adapt our model to their needs (**control**) and choose from different levels of abstraction **(flexibility)**. Moreover, easy access and high **usability** are critical factors for the adoption of any new proposal. Therefore, we aim for a solution that requires little time for setup and parameterisation. Furthermore, we want to facilitate the **repeatability** of experiments i. e., it should be easy for researchers to share their experimental setups with the scientific community. Since there is no ultimate evaluation platform (cf. Sect. 2) and we cannot implement our proposal for all platforms, we want to assert easy **adaptability**.

3.3 Selecting a Suitable Workload Modelling Approach

Traffic generators can be classified according to their insertion level into application-level, flow-level (TCP) and packet-level (IP) generators. We find **application-level generators** to be the most appropriate: Packet generators (probably the most common type) and flow generators are not as appropriate because anonymity networks do not directly forward IP packets or TCP flows for both performance (overhead for establishing channels) and security reasons (hiding the number of real connections). Among the application-level

workload models, we considered **two common approaches** for our solution: **Application-Specific Models** (ASM, cf., e. g., [4, 8]), which try to model user or application behaviour itself (e. g., via state machines) and **Extraction-Based Models** (EBM), that try to extract application behaviour from packet header traces recorded in real networks (cf., for instance, [2, 16, 26]).

While **ASMs** provide a higher level of control and accuracy, they also require a separate model for each application of interest (increasing complexity) and they require adaptation when application behaviour changes, e. g., when new protocols like [6] gain currency (diminishing flexibility). Moreover, the experimenter has to choose realistic values or distributions for several parameters (flexibility vs. usability). Those values are typically derived from trace files or previous studies. **EBMs** are more flexible as they are not tailored to a single application's behaviour. The level of detail achievable with EBMs is lower, though, since the packet traces required by these models (and provided by different research institutes, e. g., [21, 24]) are typically truncated after the transport layer header for anonymity and storage reasons. Therefore, some details, like whether a transmitted data block contains a single HTTP response or several HTTP responses sent within a short time frame, cannot be reconstructed (reducing accuracy). However, EBMs provide better usability, as most parameters that have to be configured by the experimenter in ASMs can be automatically derived from the source trace files in EBMs.

Due to usability advantages and implicit support for different applications, we decided to implement an EBM for our purposes.

3.4 The Dependency-Preserving Extraction Model

To extract an application-neutral characterisation of host behaviour from a packet header trace, both **models for individual flows** and **models for the relations between flows** are required. For this purpose we extend the *A-B-T Model*, the standard model of ns-2 and ns-3 [26].

The basic idea of the *A-B-T Model* is to *reverse-engineer* the read and write operations of applications from a packet header trace. To this end, an analysis of the sequence and acknowledgement numbers of TCP packets is performed to infer the size of data units transferred on the application layer (Application Data Units, ADUs). This information is stored in so-called *connection vectors*. Each vector consists of n epochs. An epoch is a triplet of a request size A, a reply size B and a delay T between epochs (cf. Fig. 3). The payload length of consecutive packets is interpreted as a single ADU until a packet in the opposite direction is received (starting a new epoch). The delay is derived from packet timestamps.

One problem with the *A-B-T Model* for our purposes is the fact that it assumes the simulation to replay ADUs with an accurate model of the TCP stack, including the simulation of the TCP feedback loop (congestion avoidance algorithms). While this approach offers a high level of detail, it also results in a strong increase of complexity for both experimental setup and runtime and may prevent medium or large scale experiments. Since we want to give experimenters the choice to simulate all individual connections or to preserve

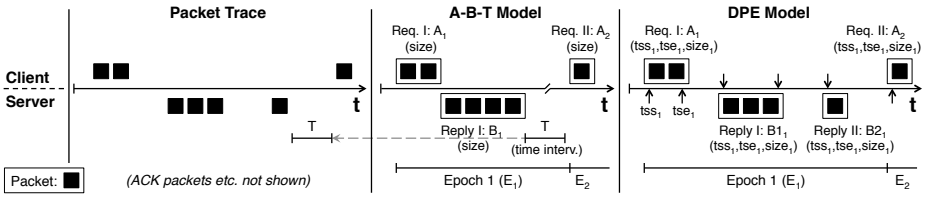


Fig. 3. From packet header trace files to the *A-B-T* and *DPE* representation

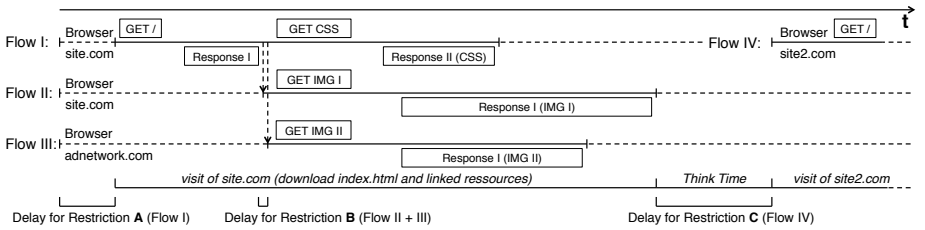


Fig. 4. Relations between the flows of a host (for the case of HTTP)

the transfer durations of the source trace (cf. Sect. 3.5), we extend the A-B-T model to **store timestamps** for the start (tss_j) and end (tse_j) of each ADU and further regard consecutive packets with a distance of more than $\tau = 1\,\text{ms}$ as individual ADUs (cf. Fig. 3).The actual value of τ can be changed by the experimenter. More formally, an epoch e in our model may contain i replies (instead of the single reply size B in the A-B-T model), represented by the triplet $r_i = (\text{tss}_i, \text{tse}_i, \text{size}_i)$. T is no longer present in our model as it (as well as all other delays between ADUs) can be computed from the absolute timestamps.

The **second problem** with the *A-B-T Model* is that it does not capture **relations between flows** [26]. Figure 4 illustrates this issue for the example of HTTP. When a modern web browser downloads a web page, it will open a single connection to request the root (HTML) document. After the arrival of the document, the browser will typically open additional connections to download referenced objects like images or CSS files. In order to preserve relations between flows, we store source and destination addresses of the hosts involved and use the absolute timestamps of our extended epoch representation (see above) to calculate restrictions between flows of the same host.

A **restriction** is bound to a flow and contains a *target event* and a *delay*. A flow may not be replayed in the testbed before the *target event* occurred in the simulation and the additional *delay* has passed. If several flows are open at the same time, the *target event* will be the latest finished reply of a parallel flow, as the new flow might have been established due to that reply (cf. *Restriction B* in Fig. 4). If no open flows have received a reply yet, the *target event* will be the end of the latest finished flow (*Restriction C* in Fig. 4) or the start of the trace file if no flows are finished yet (*Restriction A*). The *delay* is simply the offset of the flow in question from the *target event* as observed in the source trace.

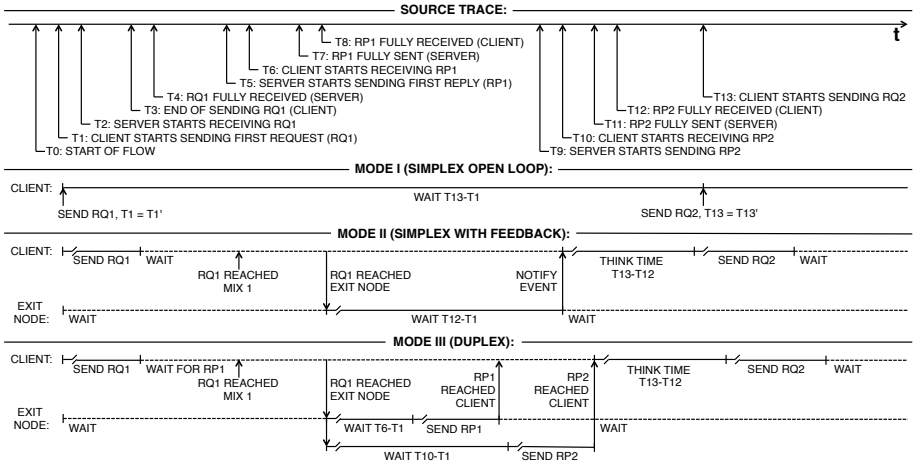


Fig. 5. Replay modes of the R&F Model

3.5 The Replay and Feedback (R&F) Model for Load Generation

The R&F Model determines how the flows extracted by the DPE Model are replayed. It supports different levels of detail that affect realism, control and complexity. One of **three replay modes** (two simplex and one duplex mode) can be selected. We will explain each mode for the example of a transaction between a client and a server via HTTP (cf. Fig. 5).

Mode 1 simply replays requests in an **open simplex loop**, i. e., the simulated clients use a fixed schedule and send each request at the same simulated time (T1′ and T13′ in Fig. 5) as in the source trace (T1 and T13), i. e., $T1 = T1'$ and $T13 = T13'$. This mode is used in [25] and reflects common assumptions of analytic evaluations. *Mode 1* is useful to understand the basic properties of the object of study as results are not blurred by other effects. These properties as well as correlations between involved parameters are difficult to derive from more detailed and realistic evaluations. However, using Mode 1 will still be more realistic than, for instance, modelling the arrival of messages by a poisson process.

Mode 1 has two significant limitations: It should only be used when connections are modelled with *unlimited bandwidth* as otherwise the connections between clients and first mix (*C–M*, cf. Fig. 2) may become the bottleneck, which reduces the burstiness of flows (sending buffers of clients will most or all of the time be filled with requests) [13]. Even with unlimited bandwidth, requests that are dependent on previous replies (e. g., RQ2 at T13 in Fig. 5 might have been caused by RP1 at T12) might be replayed before the reply in question has reached the simulated server, i. e., the delay introduced by the anonymity system does not affect the simulated sending behaviour of clients.

The remaining two replay modes take feedback from the system under test into account to prevent these effects (*closed loop*). All modes assert that the *think time* (T13 − T12) between requests is always preserved.

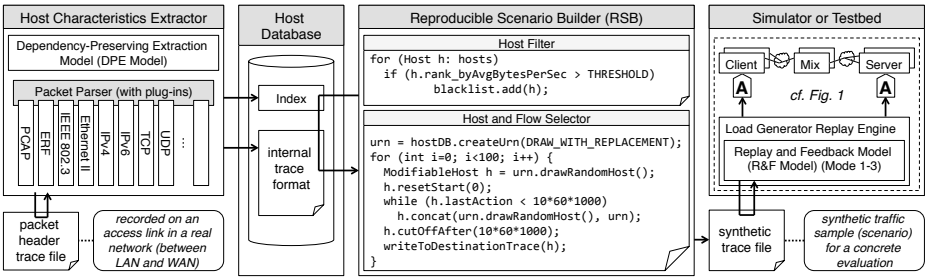


Fig. 6. Architecture of our evaluation suite

In **Mode 2**, the client will wait after sending a request until a *notify event* is observed. The purpose of the notify event is to ensure that the delays introduced by communication channels (e. g., *C–M* and *M–M* connections) and by the mixes themselves can be modelled. The *notify event* is triggered by the *Exit Node* after it has received all replies for the client's request (*Exit Nodes* run a proxy that requests data from servers on client's behalf (cf. Fig. 2). The client will send its next request (RQ2 in Fig. 5) only after the *notify event* has been observed and the additional think time has passed. The **assumption** in this model is, that servers are able to answer requests in the same time as observed in the original trace. While this is a simplifying assumption, its effect on accuracy should be small as the delay introduced inside the anonymity network (through delaying messages to build an anonymity set or congestion) is usually the bottleneck, i. e., *Exit Nodes* will typically be able to receive data from servers much faster than they can forward them through the anonymous reply channels to clients.

In **Mode 3** (duplex mode) clients will wait after sending a request until they receive the corresponding reply (or replies) and (after the additional think time) send the next request (RQ2 in Fig. 5). In this mode the delays introduced by communication channels and mixes can be modelled for both requests and replies. *Exit Nodes* will start forwarding replies to clients as soon as they receive the first bytes (of theses replies) from the corresponding servers ($T6 - T1$ and $T10 - T1$ in Fig. 5). As in *Mode 2*, delays for these incoming replies on *Exit Nodes* are recreated as in the original trace. *Mode 3* offers the highest level of detail among the three modes and allows predictions about user-perceived quality of service attributes (e. g., RTT and throughput).

4 Implementation of our Workload Model

We have implemented the DPE and R&F workload models described in Sect. 3 and integrated them into an evaluation suite that can be re-used by others. The evaluation suite (cf. Fig. 6) consists of **four main components**: the Host Characteristics Extractor, the Host Database, the Reproducible Scenario Builder, and a Simulator or Testbed.

The **Host Characteristics Extractor (HCE)** is the implementation of the DPE Model. Its input consists of a packet header trace file recorded in a real network. The HCE uses packet parsers (e. g., for PCAP and ERF and higher-level protocols) to extract an application-level characterisation for each host from the trace file that consists of flows and restrictions according to the DPE Model. Additionally, aggregated statistics (see below) are recorded for each host. The output of the HCE is stored in an intermediate format in the Host Database.

Building workload models typically involves selecting a portion of hosts that meet some desired criteria from the raw trace files. This is problematic as the size of suitable trace files is typically much higher than the available RAM (e. g., we use a 23 GB sample from the 2009 "Auckland 10" data set [24] for our evaluation in Sect. 5). The **Host Database** is an efficient solution for that task. Compared to the approach typically encountered in the network research community, namely iteratively traversing the whole trace with packet filters [3], our solution is faster and more flexible: During parsing the HCE records *aggregated statistics* for various behavioral characteristics for each host and stores them in the Host Database. Inspired by *Information Retrieval* systems the Host Database creates an *index* that allows for fast selection of the traffic of those hosts that meet certain selection criteria for a concrete experiment. At the moment the index contains about 30 statistics, among them the average sending rate and number of flows for each host. Furthermore, it contains the ranks of hosts for each attribute. As a result, it is easy to perform data cleansing (e. g., blacklisting the 5 % hosts with the highest sending rate) and to select adequate hosts for a realistic test scenario. The Host Database is used by the Reproducible Scenario Builder to create synthetic trace files that represent concrete evaluation scenarios based on the sending and receiving behaviour of hosts.

The **Reproducible Scenario Builder (RSB)** allows researchers to create or re-create traffic traces used for replay during simulation. This involves the selection of appropriate hosts from the *Host Database* as well as data cleansing tasks. As there is no one-fits-all approach for these tasks, we require the experimenter to specify his decisions by implementing a **Host and Flow Selector**. A typical selector fits into (much) less than 100 lines of code and can be implemented in a few minutes. Figure 6 shows a code example. Thus the effort for implementing the selector should be almost negligible compared to the decision process required to define an adequate scenario (usability). If an experimenter publishes his extractor and states his input trace file, other scientists can recreate the same synthetic output trace file (repeatability). We include several *standard selectors* that address typical evaluation scenarios, e. g., selectors that choose n random hosts that are continuously online for a duration of m minutes, selectors that only take into account specific protocol mixes (e. g., HTTP and HTTPs only), as well as selectors for x hosts with a *high* sending rate and y hosts with a *low* sending rate.

The RSB can create both *unmodified* and *re-composed* workload sets. While unmodified replay of host characteristics is preferable in terms of realism and accuracy, re-composed workload sets allow for more control and may be more

suitable to identify, understand and verify correlations. To this end, selectors may make use of several methods that allow to change the characteristics of a host. For instance, *offline phases* (i. e., periods without data transfer with a minimum length of y ms) may be removed and flows of different hosts can be concatenated or cut off. Furthermore, various random samples, e. g., think times, can be drawn from the index. While re-composed workloads cannot offer the same level of control as analytical or probabilistic traffic models, they offer a noticeable increase of flexibility compared to unmodified extraction from traces.

The synthetic trace files generated by the RSB are used as input for the Load Generator of the Simulator or Testbed component, in which the experiments are carried out. Based on the application-level characterisation stored in the synthetic trace the **Load Generator** simulates individual clients (implementing the R&F Model). It interacts with a **Simulator or Testbed** that represents the system under test. Typical experiments supported by the evaluation suite include: (1) evaluations of the overhead introduced by a certain anonymity system against a baseline, i. e., the quality of service attributes measured in the source trace, (2) comparisons of the performance of different anonymity system proposals, (3) validations of the severity of different traffic analysis attacks [19], and (4) finding the suitable parameters for an anonymity system proposal.

In principle, our evaluation suite can be used to evaluate any low-latency anonymity system. We provide an implementation for the discrete-event network simulator of *gMix* (cf. Sect. 1), because it is available as open source software and already includes abstract models for several mix types. However, we expect easy adaptability for other platforms (cf. Sect. 2) as solely a replay engine capable of parsing our synthetic trace files is required (only a small fraction of the 10,000 SLOC in total). The HCE, the Host Database and the RSB can be re-used.

5 Evaluation

Our evaluation serves two purposes: firstly, we validate the accuracy of our two complementary models and their implementations, and secondly, in order to show the importance of realistic workload models for anonymity systems, we compare characteristics of the traffic created by our dependency-preserving workload model with the traffic created by the more simplistic extraction technique used in [25]. All source code and configuration files can be downloaded from the project website (cf. Sect. 1), including details on how to reproduce our results.

In order to **validate the accuracy** of our workload model we collect traffic characteristics in the source trace files and compare the obtained values with the values computed for traffic being replayed in a simulation. A similar methodology has been used for the evaluation of the A-B-T model [16]. As we focus our attention on the workload model in this experiment, the measurements are performed for a simulated anonymity system that introduces no delay with all connections having unlimited bandwidth. We used Mode 3 of the R&F Model (duplex) for this evaluation. Figure 7 (left-hand side) shows the resulting cumulative distribution function of the characteristic "ADU sizes" for two samples

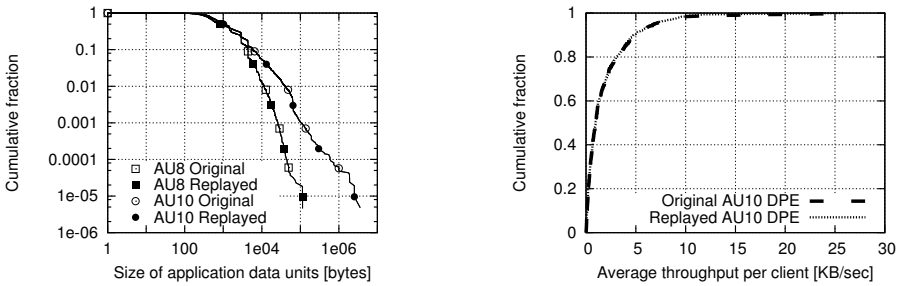


Fig. 7. Evaluation of the accuracy of our workload model

from the data sets, "Auckland 8" (2003) and "Auckland 10" (2009). Figure 7 (right-hand side) displays results for the characteristic "average throughput per client". According to these (and several other, not shown) measurements, traffic replayed using our workload model does not exhibit any significant differences in comparison to the source traces.

Finally, we illustrate the **relevance of realistic workload models** for the prediction of the behaviour of anonymity systems. For this purpose we present a case study in which we evaluate the behaviour of the DLPA dummy traffic scheme [25] for two different workload models, namely, *DPE* (our model) and *DLPAE*. DLPAE implements the extractor used in [25]: this extractor simply *picks up* and concatenates flows from the source trace for each client. Figure 8 (left-hand side) shows that the average throughput per client generated by DLPAE is considerably higher than the throughput generated by DPE. This result is due to the fact that traffic formed by concatenating flows is not as bursty as the real traffic. In the case of DPE, periods of inactivity (*think times*, cf. Sect. 4) reduce the average throughput.

The DLPA dummy traffic scheme has been only assessed with DLPAE so far [25]. As the efficiency of a dummy traffic scheme depends on the sending behaviour of the clients, its suitability for real-world, bursty traffic is questionable. We have investigated this hypothesis by measuring the amount of dummy messages that is output by a DLPA node (mix). The simulated mix exerts a maximum processing delay of $\Delta = 1$ second. We find that for 100 users 86 % of the output messages are dummies when traffic is modelled with DLPAE (cf. graph on the right-hand side of Fig. 8). With the more realistic traffic generated by our DPE/R&F Model, only 10 concurrent users can be handled by the mix to achieve a similar efficiency. This difference can be explained by the think times that dominate the real client behaviour. Accordingly, the efficiency of DLPA can be expected to be significantly worse in practice than estimated previously.

While we have only demonstrated the importance of realistic workload modelling for the DLPA, it is certainly of interest for other low-latency anonymity systems as well. Our models and tools can be used to reduce the complexity of this task. We hope that our contribution motivates other researchers to evaluate existing and novel proposals with realistic workloads.

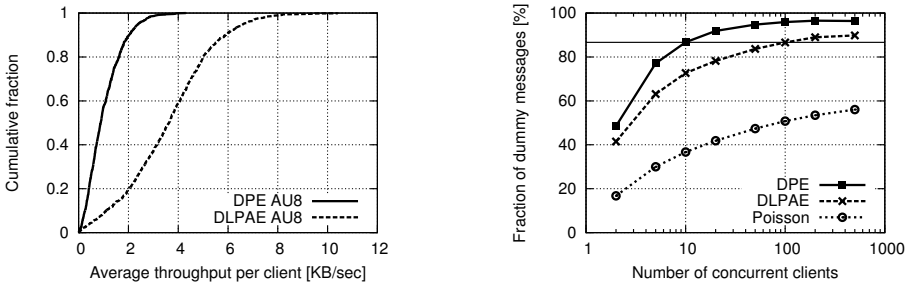


Fig. 8. Comparison of our workload model with previous work

6 Conclusion

Evaluating a distributed system thoroughly is a laborious task, which entails many critical decisions. Nevertheless, an empirical evaluation of anonymity systems is essential to understand the factors that influence their performance in practice. Unfortunately, for many proposed and practical systems there has been little work on comparable, repeatable and realistic evaluations so far.

Our work serves two purposes: Firstly, we strive to provide a usable, more realistic workload model that can be employed in simulations to predict the attainable performance of a system in a real-world setting. In contrast to previous work, we ensure that dependencies between flows are maintained during the simulation, which allows us to mimic the real behaviour of applications more closely. Secondly, we want to work towards a standardised evaluation methodology for the evaluation of anonymity systems that reduces upfront efforts and ensures repeatability of experiments. We believe our evaluation suite and the included scenario builder are first steps in that direction.

Acknowledgments We thank our colleague Andrey Kolesnikov (Telecommunications and Computer Networks Group) for insightful discussions regarding workload modelling in the network research community.

References

1. Agrawala, A., Mohr, J., Bryant, R.: An Approach to the Workload Characterization Problem. Computer 9(6), 18–32 (Jun 1976)
2. Alcock, S., Lawson, D., Nelson, R.: Extracting Application Objects from TCP Packet Traces. In: Australasian Telecommunication Networks and Applications Conference. pp. 151–156 (2007)
3. Alcock, S., Lorier, P., Nelson, R.: Libtrace: A Packet Capture and Analysis library. SIGCOMM Comput. Commun. Rev. 42(2), 42–48 (2012)
4. Barford, P., Crovella, M.: Generating Representative Web Workloads for Network and Server Performance Evaluation. In: SIGMETRICS. pp. 151–160 (1998)
5. Bauer, K., Sherr, M., McCoy, D., Grunwald, D.: ExperimenTor: A Testbed for Safe Realistic Tor Experimentation. In: Workshop on Cyber Security Experimentation and Test (2011)

6. Belshe, M., Peon, R., Thomson, M., Melnikov, A.: SPDY Protocol. Internet Draft (2012), http://tools.ietf.org/html/draft-ietf-httpbis-http2-00
7. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A System for Anonymous and Unobservable Internet Access. In: Federrath [12], pp. 115–129
8. Cao, J., Cleveland, W.S., Gao, Y., Jeffay, K., Smith, F.D., Weigle, M.C.: Stochastic Models for Generating Synthetic HTTP Source Traffic. In: INFOCOM (2004)
9. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24(2), 84–90 (1981)
10. Dhungel, P., Steiner, M., Rimac, I., Hilt, V., Ross, K.W.: Waiting for Anonymity: Understanding Delays in the Tor Overlay. In: Peer-to-Peer Computing. pp. 1–4. IEEE (2010)
11. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: 13th USENIX Security Symposium. pp. 303–320 (2004)
12. Federrath, H. (ed.): Designing Privacy Enhancing Technologies, Proceedings, LNCS, vol. 2009. Springer (2001)
13. Floyd, S., Paxson, V.: Difficulties in Simulating the Internet. IEEE/ACM Trans. Netw. 9(4), 392–403 (2001)
14. Fuchs, K.P., Herrmann, D., Federrath, H.: Introducing the gMix Open Source Framework for Mix Implementations. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012, LNCS, vol. 7459, pp. 487–504. Springer (2012)
15. Hernández-Campos, F., Jeffay, K., Smith, F.D.: Tracking the Evolution of Web Traffic: 1995–2003. In: MASCOTS. pp. 16–25. IEEE (2003)
16. Hernández-Campos, F., Smith, F.D., Jeffay, K.: Generating Realistic TCP Workloads. In: Int. CMG Conference. pp. 273–284. (2004)
17. Jansen, R., Hopper, N.: Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In: Proceedings of the Network and Distributed System Security Symposium (NDSS'12). Internet Society (2012)
18. McCoy, D., Bauer, K., Grunwald, D., Kohno, T., Sicker, D.: Shining Light in Dark Places: Understanding the Tor Network. In: Borisov, N., Goldberg, I. (eds.) Privacy Enhancing Technologies, LNCS, vol. 5134, pp. 63–76. Springer (2008)
19. Raymond, J.F.: Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In: Federrath [12], pp. 10–29
20. Sreeram Ramachandran: Web metrics: Size and number of resources (May 2010), https://developers.google.com/speed/articles/web-metrics
21. The Cooperative Association for Internet Data Analysis: http://www.caida.org/
22. Vishwanath, K.V., Vahdat, A.: Swing: Realistic and Responsive Network Traffic Generation. IEEE/ACM Trans. Netw. 17(3), 712–725 (2009)
23. Wacek, C., Tan, H., Bauer, K., Sherr, M.: An Empirical Evaluation of Relay Selection in Tor. In: Proceedings of the Network and Distributed System Security Symposium (NDSS'13). Internet Society (2013)
24. WAND Network Research Group: http://www.wand.net.nz/wits/
25. Wang, W., Motani, M., Srinivasan, V.: Dependent Link Padding Algorithms for Low Latency Anonymity Systems. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM Conference on Computer and Communications Security. pp. 323–332. ACM (2008)
26. Weigle, M.C., Adurthi, P., Hernández-Campos, F., Jeffay, K., Smith, F.D.: Tmix: A Tool for Generating Realistic TCP Application Workloads in ns-2. Computer Communication Review 36(3), 65–76 (2006)
27. Wendolsky, R., Herrmann, D., Federrath, H.: Performance Comparison of Low-Latency Anonymisation Services from a User Perspective. In: Borisov, N., Golle, P. (eds.) Privacy Enhancing Technologies, LNCS, vol. 4776, pp. 233–253. Springer (2007)