# LDplayer: DNS Experimentation at Scale (abstract with poster)

USC/ISI Technical Report ISI-TR-721, August 2017*

Liang Zhu
USC/Information Sciences Institute

John Heidemann
USC/Information Sciences Institute

## 1 INTRODUCTION

The Domain Name System (DNS) has grown to play various of broader roles in the Internet, beyond name-to-address mapping. It provides query engine for anti-spam [3] and replica selection for content delivery networks (CDNs) [4]. DANE [2] provides additional source of trust by leveraging the integrity verification of DNSSEC [1]. The wide use and critical role of DNS prompt its continuous evolution.

However, DNS protocol evolution and expansion of its use have been slow because advances must consider a huge and diverse installed base: a complex ecosystem of many implementations, archaic deployments, and interfering middleboxes.

DNS performance issues are also a concern, both for choices about protocol changes, and for managing inevitable changes in use. There are a number of important open questions: How does current server operate under the stress of a Denial-of-Service (DoS) attack? What is the server and client performance when protocol or architecture changes?

We believe accurate, high-speed *trace replay* is essential to study many open questions in DNS, because DNS performance can be very sensitive to query timing and caching, and interactions across levels of the DNS hierarchy and multiple servers. These interactions seem impossible to model, and difficult to capture with a naive set of servers.

In this poster we will describe *LDplayer*, a configurable, general-purpose DNS testbed that enables DNS experiments at scale in several dimensions: many zones, numerous levels of DNS hierarchy, large query rates, and diverse query sources. To meet these requirements while providing high fidelity experiments, LDplayer includes a distributed DNS query replay system and methods to rebuild the relevant DNS hierarchy from traces. We show that a single DNS server can correctly emulate multiple independent levels of the DNS hierarchy while providing correct responses as if they were independent. We show the importance of our system to evaluate pressing DNS design questions, using it to evaluate changes in DNSSEC key size.

## 2 LDPLAYER: DNS TRACE PLAYER

We next list the high-level design requirements for our system (§2.1) and the architecture to meet these requirements (§2.2).

### 2.1 Design Requirements

The goal of LDplayer is to provide a controlled testbed for repeatable experiments upon realistic evaluation of DNS performance. To meet this goal, we must achieve the following important requirements.

**Emulate complete DNS hierarchy efficiently:** LDplayer must emulate multiple independent levels of the DNS hierarchy and provide correct responses using minimal commodity hardware. It is not scalable to use separated servers or virtual machines to host each zone because of hardware limits and many different zones in a network trace. A single server providing many zones of DNS hierarchy does not work directly, because the server gives the final DNS answer straightly and skips the round trip of DNS referral replies.

**No replayed traffic leakage to the Internet:** Experimental traffic must stay inside the testbed, without polluting the Internet. Otherwise each experiment could leak bursts of requests to the real Internet, and simulations of high rates or parallel experiments might stress real-world servers.

**Repeatability of experiments:** LDplayer needs to support repeatable and controlled experiments. For different experiment trials, the replies to the same set of query replay should stay the same. This reproducibility is very important for experiments that require fixed query-response content to evaluate new transform in DNS, such as protocol changes and new server implementations.

**Enable experiments with traffic variations:** Replay must be able to manipulate traces to answer "what if" questions with variations of real traffic. Since input is normally binary network trace files, the main challenge is how to provide a flexible and user-friendly mechanism for query modification. We must minimize the delay by query manipulation, so that input processing is fast enough to keep up with real time.

**Accurate timing at high query rates:** LDplayer must be capable of replaying queries at fast rates while preserving correct timing, and reproduce real-world traffic patterns for both regular and under attack. However, both using a single host and many hosts have challenges. Due to resource constraints on CPU and the number of ports, a single host may not be capable to replay fast query stream or emulate diverse sources. A potential solution is to distribute input to different hosts, however, it brings another challenge in ensuring the correct timing and ordering of individual queries.

**Support multiple protocols effectively:** LDplayer needs to support both connectionless (UDP) and connection-oriented (TCP and TLS) protocols, given increasing interest in DNS over connections [7]. However, connection-oriented protocols bring challenges in trace replay: emulating connection reuse and round-trip time (RTT). The query replay system of LDplayer is the first system that can emulate connection reuse

*This poster abstract was published as part of SIGCOMM 2017 [6]. This technical report adds a copy of the poster.

for DNS over TCP; Other tools, such as Bit-Twist and Tcpreplay, replay each packet in the trace mechanically. We expect to emulate RTT based on real-world distributions.

### 2.2 Architecture

We next describe LDplayer's architecture (Figure 1). With captured network traces of DNS queries (required) and responses (optional), a researcher can use the *Zone Constructor* to generate required zones. LDplayer uses a *single* authoritative DNS server with proxies to emulate entire DNS hierarchy (*Hierarchy Emulation*). The single DNS server provides all the generated zones. The proxies manipulate packet addresses to make the authoritative server provide correct answers. As a distributed query system, the *Query Engine* replays queries in the captured traces. Optionally, the researcher can use *Query Mutator* to change the original queries arbitrarily.

Each component in LDplayer addresses a specific design requirement from §2.1. In LDplayer's zone constructor, we *synthesize data for responses* and generate required zone files by performing one-time fetch of missing records. We run a real DNS server that hosts these reusable zone files and provides answers to replayed queries, to have repeatable experiments without disturbing the Internet. By default, our system provides consistent replies. Simulating dynamic address mapping like CDN is future work.

With generated zone files, we need to *emulate DNS hierarchy* to provide correct answers. Logically, we want many server hosts, one per each zone, like the real world. However, we compress those down to a single server process with single network interface using split-horizon DNS [5], so that the system is scalable to many different zones. We redirect the replayed experimental traffic to proxies, which manipulate packet addresses to discriminate queries for different zones to get correct responses. We could run multiple instances of the server to support large query rate and massive zones.

In LDplayer's query mutator, we pre-process the query trace, so that query manipulation does not limit replay times. We convert network traces to human-readable plain text for flexible and user-friendly manipulation. We then convert changed query text to a customized binary stream of internal messages for fast query replay. In principle, at lower query rates, we could manipulate a live query stream in near real time.

In LDplayer's query engine, we use a central controller to coordinate queries from many hosts and synchronize the time among different hosts, so that LDplayer can replay large query rates accurately. The query engine can replay queries via different protocols (TCP or UDP) effectively. We distribute queries from the same source addresses in the original trace to the same end queriers for replay, in order to emulate queries from the same sources which is critical for connection reuse. LDplayer replays queries based on the timing in the original trace without preserving query dependencies.

The software of our system will be publicly available at: https://ant.isi.edu/software/ldplayer/index.html.

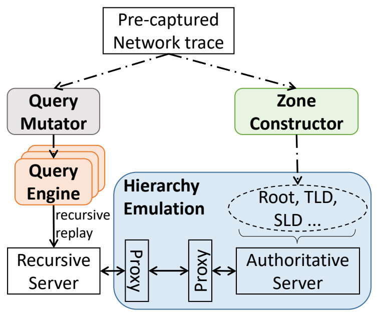


**Figure 1: LDplayer architecture**

## 3 APPLICATIONS

Our system enables applications of answering important research questions. We next present example applications.

**Impact of Change in DNSSEC Key Size:** Longer Zone Signing Key (ZSK) and more queries DNSSEC enabled (DO bit set) will increase reply traffic. By using LDplayer to replay B-Root query traffic, we evaluate scenarios with different key sizes, and different mixes (up to 100%) of DNSSEC-enabled traffic. Our experiment shows that going from 72% DO (today) to 100%, root response traffic becomes 225 Mb/s (median) with 1024-bit ZSK, and 296 Mb/s (median) with 2048-bit ZSK in steady state. Compared to 170 Mb/s with current 72% DO and 1024-bit ZSK, root response traffic could increase by 74% in the future.

**Performance of DNS over TCP and TLS:** The use of TCP and TLS improves the security and privacy of DNS. While studies have suggested increased use of TCP and TLS has only modest cost [7], trace replay can provide a more complete evaluation. Important open questions include evaluation of connection-based DNS across multiple levels of the DNS hierarchy. As a future work, we can study end-to-end client latency of DNS over TCP and TLS with RTT emulated by real-world distribution, and evaluate memory requirements on actual server implementations.

## REFERENCES

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. DNS Security Introduction and Requirements. RFC 4033. (March 2005).
[2] P. Hoffman and J. Schlyter. 2012. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698. (Aug. 2012).
[3] C. Lewis and M. Sergeant. 2012. Overview of Best Email DNS-Based List (DNSBL) Operational Practices. RFC 6471. (Jan. 2012).
[4] Ao-Jan Su, David R. Choffnes, Aleksandar Kuzmanovic, and Fabián E. Bustamante. 2006. Drafting Behind Akamai (Travelocity-based Detouring) *(SIGCOMM '06)*.
[5] Wikipedia. 2017. Split-horizon DNS. (2017). https://en.wikipedia.org/wiki/Split-horizon_DNS [Online; accessed 13-July-2017].
[6] L. Zhu and J. Heidemann. 2017. LDplayer: DNS Experimentation at Scale. In *Proceedings of the SIGCOMM Posters and Demos (SIGCOMM Posters and Demos '17)*.
[7] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya. 2015. Connection-Oriented DNS to Improve Privacy and Security. In *2015 IEEE Symposium on Security and Privacy*.

# DNS Experimentation at Scale

Liang Zhu, John Heidemann
USC/Information Sciences Institute
{liangzhu, johnh}@isi.edu

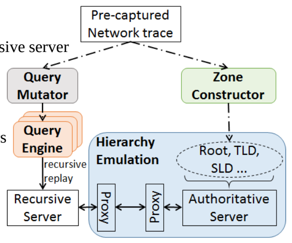
## Introduction

In the last 20 years, the Domain Name System (DNS) has grown to play various of broader roles in the Internet, beyond name-to-address mapping, such as query engine for anti-spam and replica selection for content delivery networks. However, DNS protocol evolution and expansion of its use has been slow. First, the advances of DNS must consider a huge and diverse installed base. Second, DNS performance issues are also a concern, both for choices about protocol changes, and for managing inevitable changes in use. Finally, although ideally models would guide these questions, DNS is extraordinarily difficult to model because of interactions of caching and implementation optimizations across levels of the DNS hierarchy. These motivate our work.

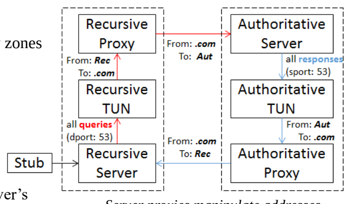
We suggest **DNS experimentation at scale** can fill this gap. We present LDplayer, a configurable, general-purpose DNS testbed. LDplayer enables DNS experiments at scale in several dimensions: many zones, multiple levels of DNS hierarchy, high query rates, and diverse query sources. To meet these requirements while providing high fidelity experiments, our approach includes a distributed DNS query replay system and methods to rebuild the relevant DNS hierarchy from traces. We show that a single DNS server can correctly emulate multiple independent DNS hierarchy while providing correct responses. We validate our system can replay DNS queries with correct timing, reproducing the DNS traffic pattern. We show the importance of our system to evaluate pressing DNS design questions, using it to evaluate changes in DESSEC key size.

## Approach

- **Construct Zone to Allow Replay Without Leakage**
  - Goal: repeatable experiment without leakage
  - Approach
    - Discover answers by replaying queries to a recursive server
    - Build zone files using captured responses
- **Emulate Whole Hierarchy Efficiently**
  - Goal: correctly answer all the replayed queries
  - Challenge: scale to 100s of zones with few computers
    - one server does not work directly
      - dump all the zones into the same server
      - *queries* to different zones are the *same*
      - *responses* from different zones are *different*
      - server does not know which zone to use
    - many servers do not work either
      - one server per zone like real world
      - run out of hardware, not scalable
  - Approach: one server with proxy
    - Single server, single address with many zones
      - Use split-horizon DNS
      - Match queries by source addresses
      - The public IP addresses of zone's nameservers as matching criteria
    - Proxies capture requests and responses
      - swap src and dst address
      - change dst address to the other server's

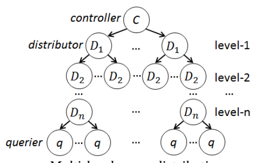


*LDplayer architecture*



*Server proxies manipulate addresses*

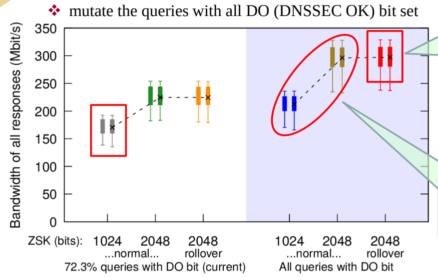
- **Support Experimental Changes of Queries**
  - Goal: flexibility in various experiments with the ability to do "What-If" evaluation
  - Approach:
    - Convert binary trace to plain text for easy manipulation
    - Convert modified text to the binary of internal messages for fast processing
- **Fast Replay of Many Queries**
  - Goal:
    - provide fast query rates
    - emulate diverse sources
    - correct timing for replayed queries
  - Challenge:
    - Resource limit in a single host:
      - CPU, memory and the number of ports
  - Approach: distribute query stream to different hosts
    - A central *Controller* managing a team of *Distributor*
    - Distributor further controls several *Queriers*



*Multi-level query distribution*

## Example Applications

- **One example**: Understand the traffic change when all queries with DNSSEC and larger key sizes
  - **Approach**:
    - Replay root DNS query traffic with different DNSSEC key sizes
    - mutate the queries with all DO (DNSSEC OK) bit set



Root response traffic could increase by 74% in the future, compared to 170Mb/s with current 72% DO and 1024-bit ZSK

With 100% DO root response traffic becomes 225Mb/s (median) with 1024-bit ZSK, and 296Mb/s (median) with 2048-bit ZSK

- **Other applications** include evaluating the performance of DNS over TCP and TLS in practice

## Motivation

- **Explore "What If" scenarios to facilitate DNS evolution**
  - Protocol and architecture changes
    - What if all DNS requests were made over TCP or TLS?
    - How does server operate under stress of a DDoS attack?
- **Hard to get definitive answers to these questions with modeling**
  - interactions of caching and implementation optimizations
  - many levels of the DNS hierarchy
- **Evolving DNS is challenging by itself**
  - Complex ecosystem
    - many implementations, deployments, and interfering middle-boxes.

accurate, high-speed trace replay
is essential to study many open questions

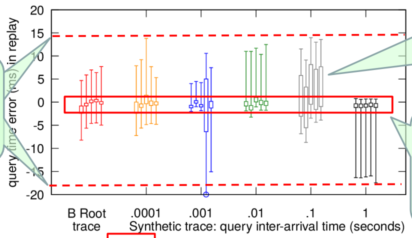
## Design Requirements

- **Emulate complete DNS hierarchy efficiently**
  - Emulate multiple independent levels of the DNS hierarchy and provides correct response, using minimal commodity hardware in a lab environment.
- **No experimental replay leakage**
  - Experimental traffic must stay inside the testbed without polluting the Internet
    - Otherwise each experiment could leak bursts of requests stressing real-world servers
- **Repeatability of experiments**
  - Experimental results should stay the same for same set of input traces
    - DNS response may change if re-looked up at experiment time
- **Flexible experiments with query mutate**
  - Replay must be able to manipulate traces to answer "what if" questions with variations of real traffic
- **Accurate timing at high query rates**
  - Replay queries at fast rates, while preserving correct timing to reproduce interesting real-world traffic patterns for both regular and under attack.
- **Support multiple protocols effectively**
  - Support both connectionless (UDP) and connection-oriented (TCP, TLS) protocol given increasing interest in DNS over connections

## Preliminary Validation

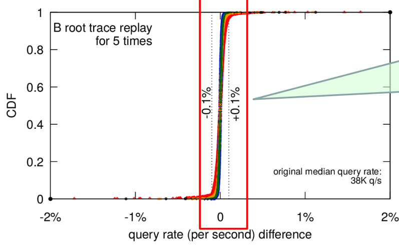
- **Goal**: validate the accuracy of our query replay system
  - Can our system reproduce the correct query timing and traffic pattern?
- **Approach**:
  - Replay different DNS traces in controlled testbed environment
  - Compare the time difference for each query and overall query rate
  - one-hour trace of a root DNS server and synthetic traces with different fixed query inter-arrival time (1 s to 0.1 ms)
- **Result**:

overall the error peek is reasonably small (±17ms )



One case with slightly larger, but still small errors (±8ms quartiles)

tiny error (mostly ±2.5ms quartiles) of query time in most cases



tiny (0.1%) difference in query rate for almost all (4 trials with 98%-99% and 1 trial with 95%) of 3.6K data points (1-hour period)

## Conclusion

- We built a configurable DNS testbed that enables DNS experiments at scale, providing a basis for DNS experimentation that can further lead to DNS evolution.
- We showed that our system can correctly emulates multiple independent levels of DNS hierarchy on a single DNS server instance.
- We validated that our query replay system can replay queries with correct timing and reproduce the traffic pattern.
- We demonstrated the power of controlled replay of traces by exploring DNS root response traffic with different DNSSEC key sizes and all queries with DNSSEC.
- Software will be available: https://ant.isi.edu/software/ldplayer