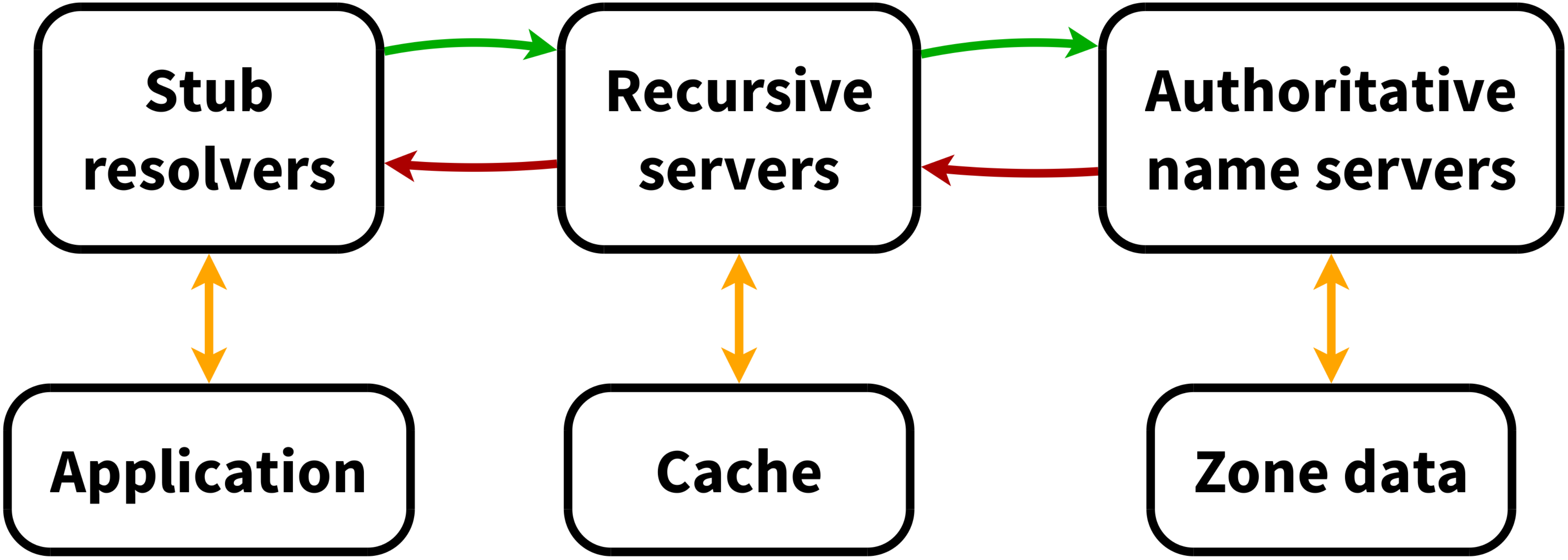
dnstap: high speed DNS logging without packet capture

Robert Edmonds (edmonds@fsi.io)
Farsight Security, Inc.

URL

- `http://dnstap.info`
  - Documentation
  - Presentations
  - Tutorials
  - Mailing list
  - Downloads
  - Code repositories

Simplified DNS overview



Stub resolvers

Recursive servers

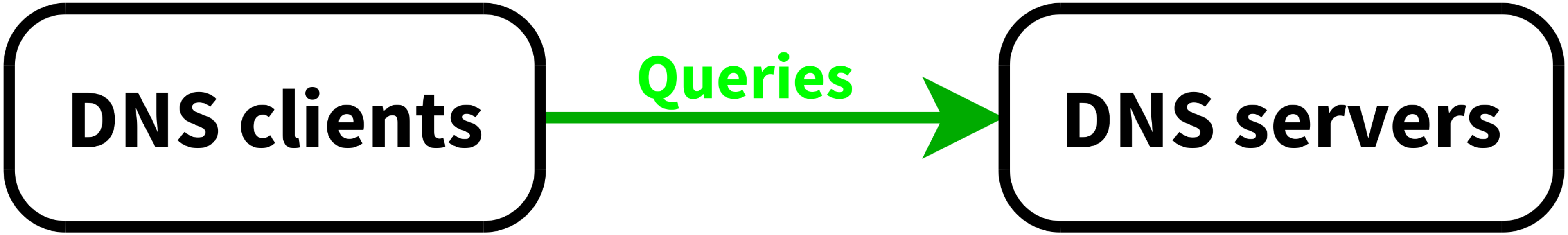
Authoritative name servers

Application

Cache

Zone data

Query logging



Query logging

- Log information about DNS **queries**:
  - Client IP address
  - Question name
  - Question type
- Other related information?
  - EDNS options
  - DNSSEC status
  - Cache miss or cache hit?
- May have to look at both **queries** and **responses**.

Query logging

- DNS server generates log messages in the normal course of processing requests.
- Reputed to impact performance significantly.
- Typical implementation:
  - Parse the request.
  - Format it into a **text string**.
  - Send to syslog or write to a log file.

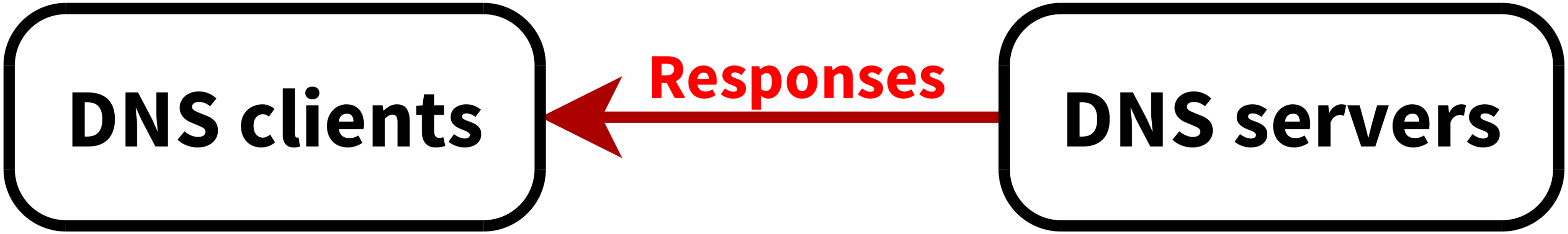
Query logging

- Implementation issues that affect performance:
  - Transforming the query into a text string takes time.
    - Memory copies, format string parsing, etc.
  - Writing the log message using **synchronous I/O** in the worker thread.
  - Using **syslog** instead of writing log files directly.
    - `syslog()` takes out a process-wide lock and does a blocking, unbuffered write for **every** log message.
  - Using stdio to write log files.
    - `printf()`, `fwrite()`, etc. take out a lock on the output stream.

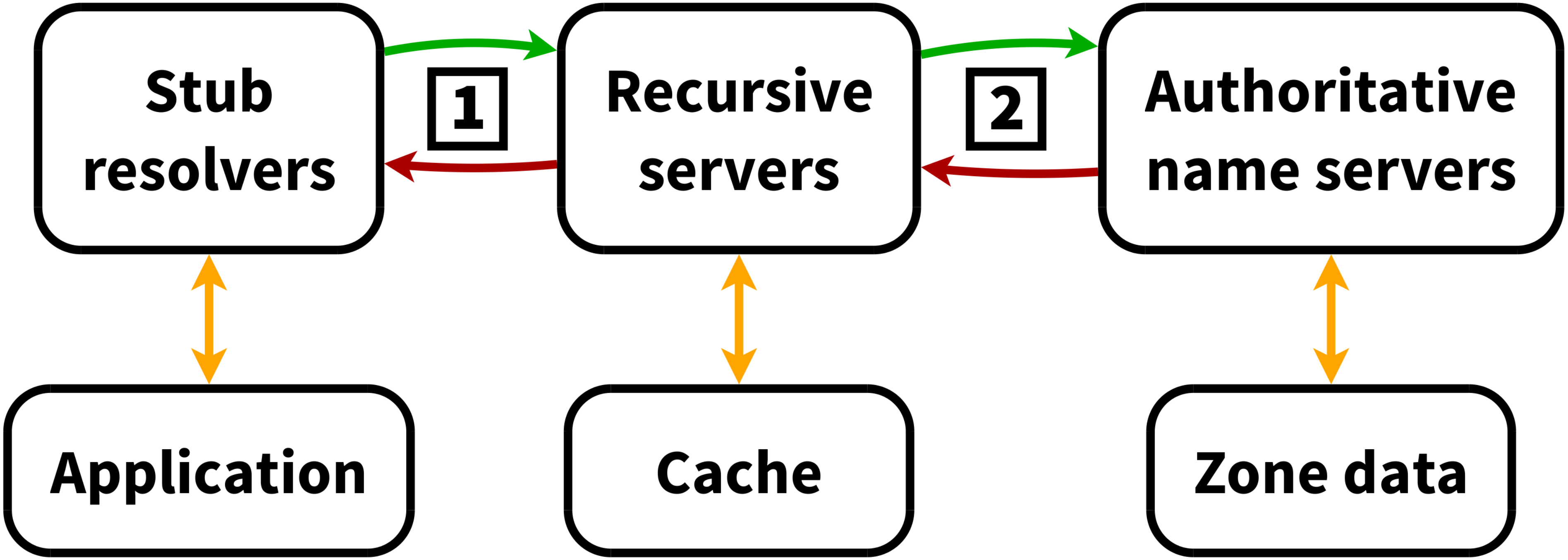
Query logging

- Do it with packet capture instead:
  - Eliminates the performance issues.
  - But, can't replicate state that doesn't appear directly in the packet.
    - E.g., whether the request was served from the cache.
- What if the performance issues in the server software were fixed?

Passive DNS replication



Passive DNS replication



- Deployment options:

  (1) “Below the recursive”

  (2) “Above the recursive”

Passive DNS replication

- Log information about **zone content**:
  - Record name
  - Record type
  - Record data
  - Nameserver IP address

Passive DNS replication

- Typical implementation:
  - Capture the DNS response packets at the recursive DNS server.
  - Reassemble the DNS response messages from the packets.
  - Extract the DNS resource records contained in the response messages.
- Low to no performance impact.

Passive DNS replication

- Issues:
  - Discard out-of-bailiwick records.
  - Discard spoofed UDP responses.
  - UDP fragment, TCP stream reassembly.
  - UDP checksum verification.
- But, the DNS server and its networking stack are already doing these things...

Insights

- Query logging:
  - Make it faster by eliminating bottlenecks like text formatting and synchronous I/O.
- Passive DNS replication:
  - Avoid complicated state reconstruction issues by capturing ***messages*** instead of ***packets***.
- Support both use cases with the same generic mechanism.

dnstap

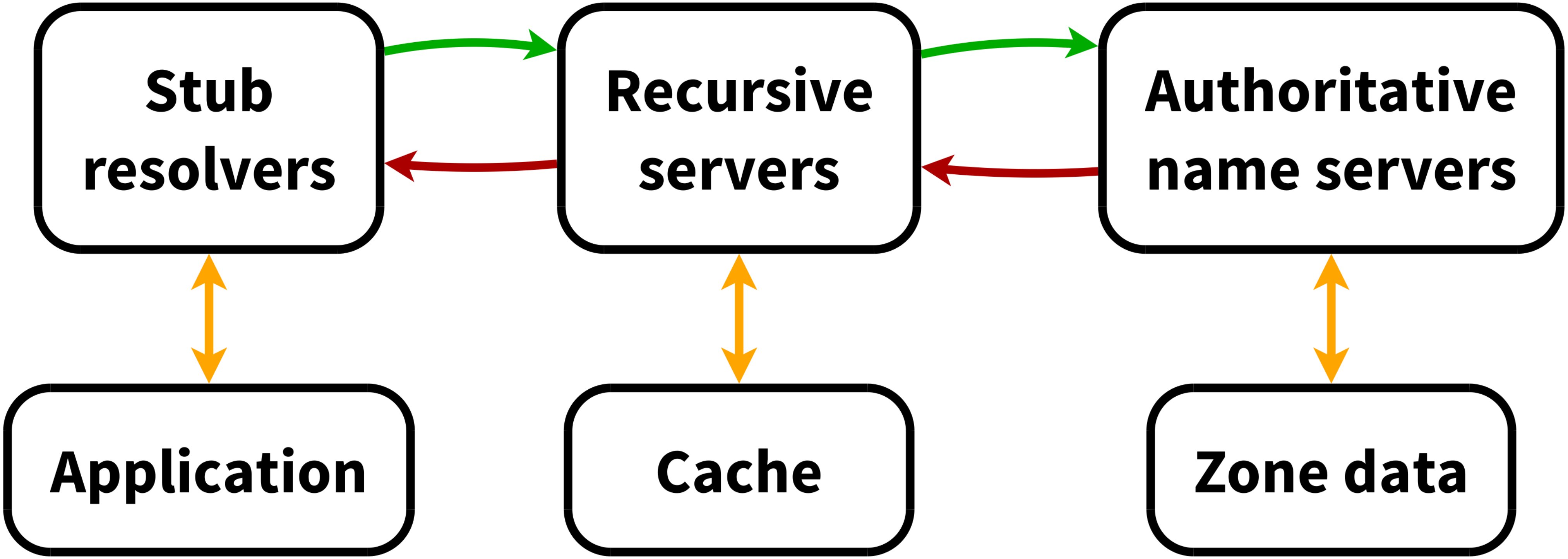
- Add a lightweight **message duplication** facility directly into the DNS server.
  - Verbatim wire-format DNS messages with context.
- Use a fast logging implementation that doesn't degrade performance.
  - Circular queues.
  - Asynchronous, buffered I/O.
  - Prefer to **drop** log payloads instead of **blocking** the server under load.

dnstap: message duplication

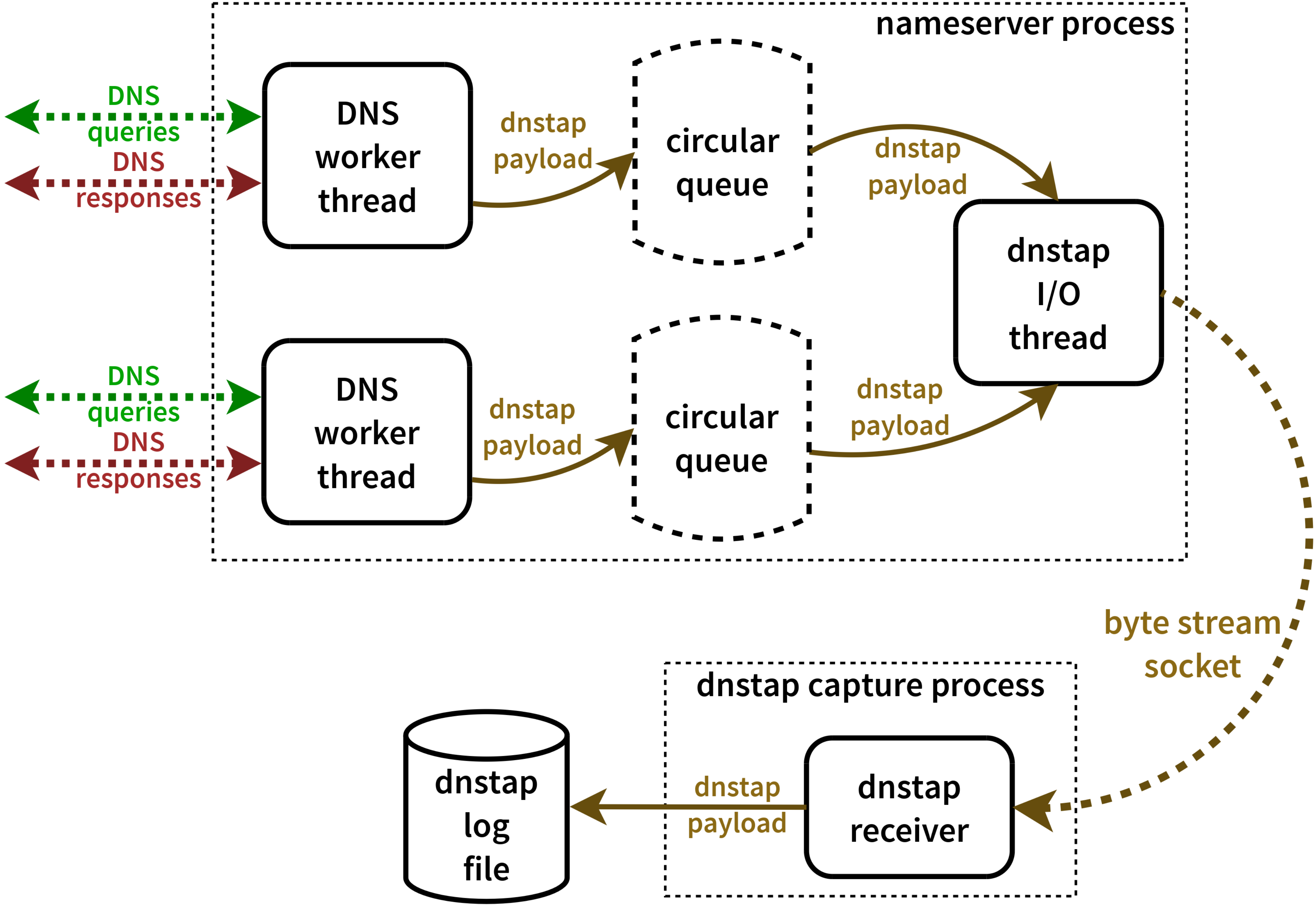
- DNS server has internal message buffers:
  - Receiving a query.
  - Sending a query.
  - Receiving a response.
  - Sending a response.
- Instrument the call sites in the server implementation so that message buffers can be duplicated and exported outside of the server process.
- Be able to enable/disable each logging site independently.

dnstap: “Message” log format

- Currently 10 defined subtypes of dnstap “Message”:
  - AUTH_QUERY
  - AUTH_RESPONSE
  - RESOLVER_QUERY
  - RESOLVER_RESPONSE
  - CLIENT_QUERY
  - CLIENT_RESPONSE
  - FORWARDER_QUERY
  - FORWARDER_RESPONSE
  - STUB_QUERY
  - STUB_RESPONSE


Stub resolvers
Recursive servers
Authoritative name servers
Application
Cache
Zone data

dnstap-enabled DNS server



Query logging with *dnstap*

- Turn on **AUTH_QUERY** and/or **CLIENT_QUERY** message duplication.
  - Optionally turn on **AUTH_RESPONSE** and/or **CLIENT_RESPONSE**.
- Connect a dnstap receiver to the DNS server.

Query logging with *dnstap*

- Performance impact should be minimal.
- Full verbatim message content is available without text log parsing.

Passive DNS replication with *dnstap*

- Turn on **RESOLVER_RESPONSE** message duplication.
- Connect a dnstap receiver to the DNS server.

Passive DNS replication with *dnstap*

- Once inside the DNS server, the issues caused by being outside disappear.
  - Out-of-bailiwick records: the DNS server already knows which servers are responsible for which zones.
  - Spoofing: the DNS server already has its state table. Unsuccessful spoofs are excluded.
  - TCP/UDP packet issues: already handled by the kernel and the DNS server.

dnstap **components**

- Flexible, structured ***log format*** for DNS software.
- ***Helper libraries*** for adding support to DNS software.
- Patch sets that ***integrate*** dnstap support into existing DNS software.
- ***Capture tools*** for receiving dnstap messages from dnstap-enabled software.

dnstap **log format**

- Encoded using Protocol Buffers.
  - Compact
  - Binary clean
  - Backwards, forwards compatibility
  - Implementations for numerous programming languages available

Helper libraries

- **fstrm**: “Frame Streams” library.
  - Encoding-agnostic transport.
  - Adds ~1.5K LOC to the DNS server.
- **protobuf-c**: “Protocol Buffers” library.
  - Transport-agnostic encoding.
  - Adds ~2.5K LOC to the DNS server.

dnstap integration

- Plans to add dnstap support to software that handles DNS messages:
  - DNS servers: BIND, Unbound, Knot DNS, etc.
  - Analysis tools: Wireshark, etc.
  - Utilities: dig, kdig, drill, dnsperf, resperf
  - More?

dnstap integration

- Unbound DNS server with dnstap support.
  - Supports the relevant dnstap “Message” types for a recursive DNS server:
    - **{CLIENT,RESOLVER,FORWARDER}_{QUERY_RESPONSE}**
  - Adds <1K LOC to the DNS server.

dnstap capture tool

- Command-line tool/daemon for collecting dnstap log payloads.
  - Print payloads.
  - Save to log file.
  - Retransmit over the network.
- Similar role to tcpdump, syslogd, or flow-tools.

Benchmarks

- More of a “microbenchmark”.
- Meant to validate the architectural approach.
- Not meant to accurately characterize the performance of a dnstap-enabled DNS server under “realistic” load.

Benchmarks

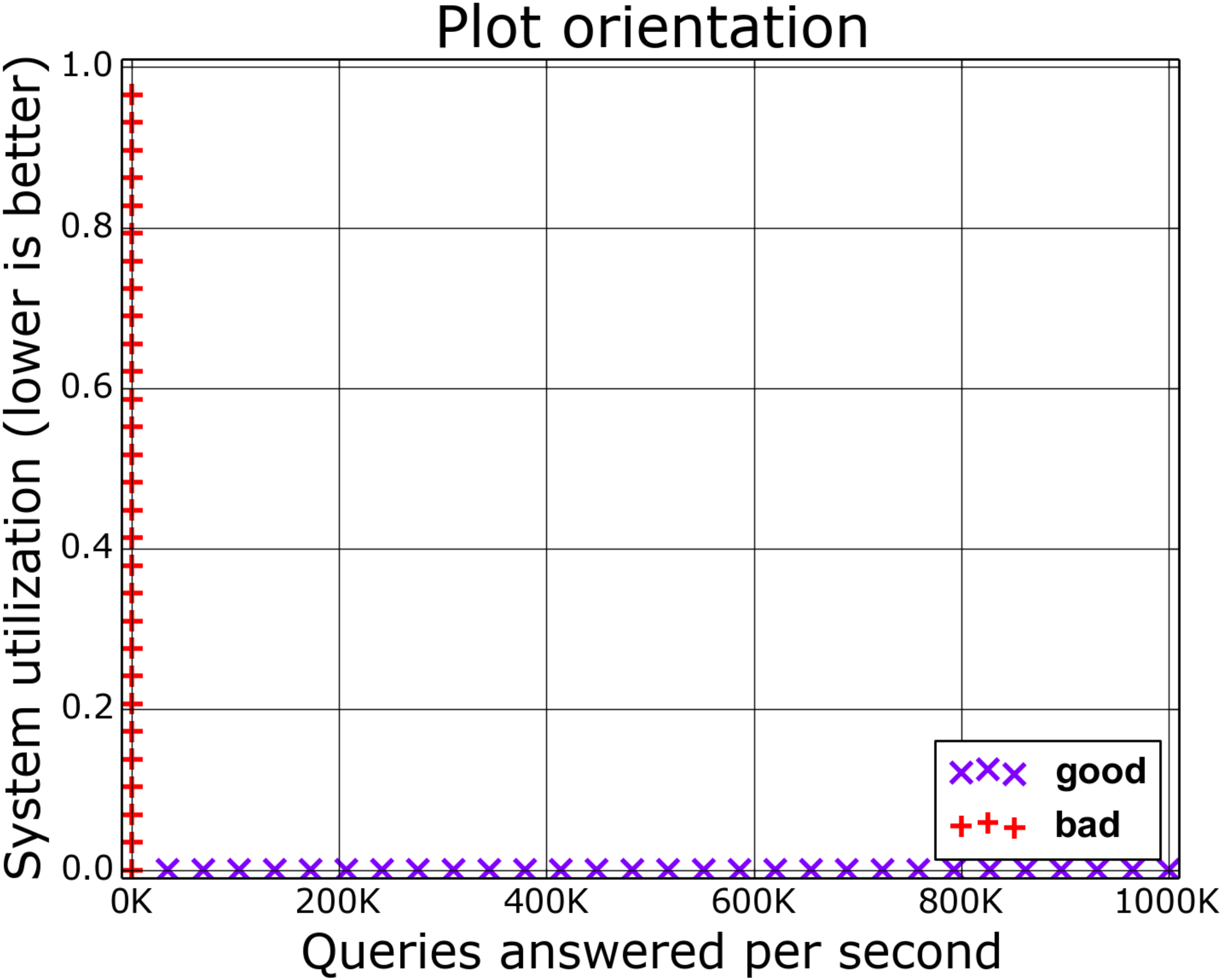
- One receiver:
  - Intel(R) Xeon(R) CPU E3-1245 v3 @ 3.40GHz
    - No HyperThreading, no SpeedStep, no Turbo Boost.
- One sender:
  - Intel(R) Core(TM) i3-4130 CPU @ 3.40GHz
- Intel Corporation I350 Gigabit Network Connection
- Sender and receiver directly connected via crossover cable. No switch, RX/TX flow control disabled.

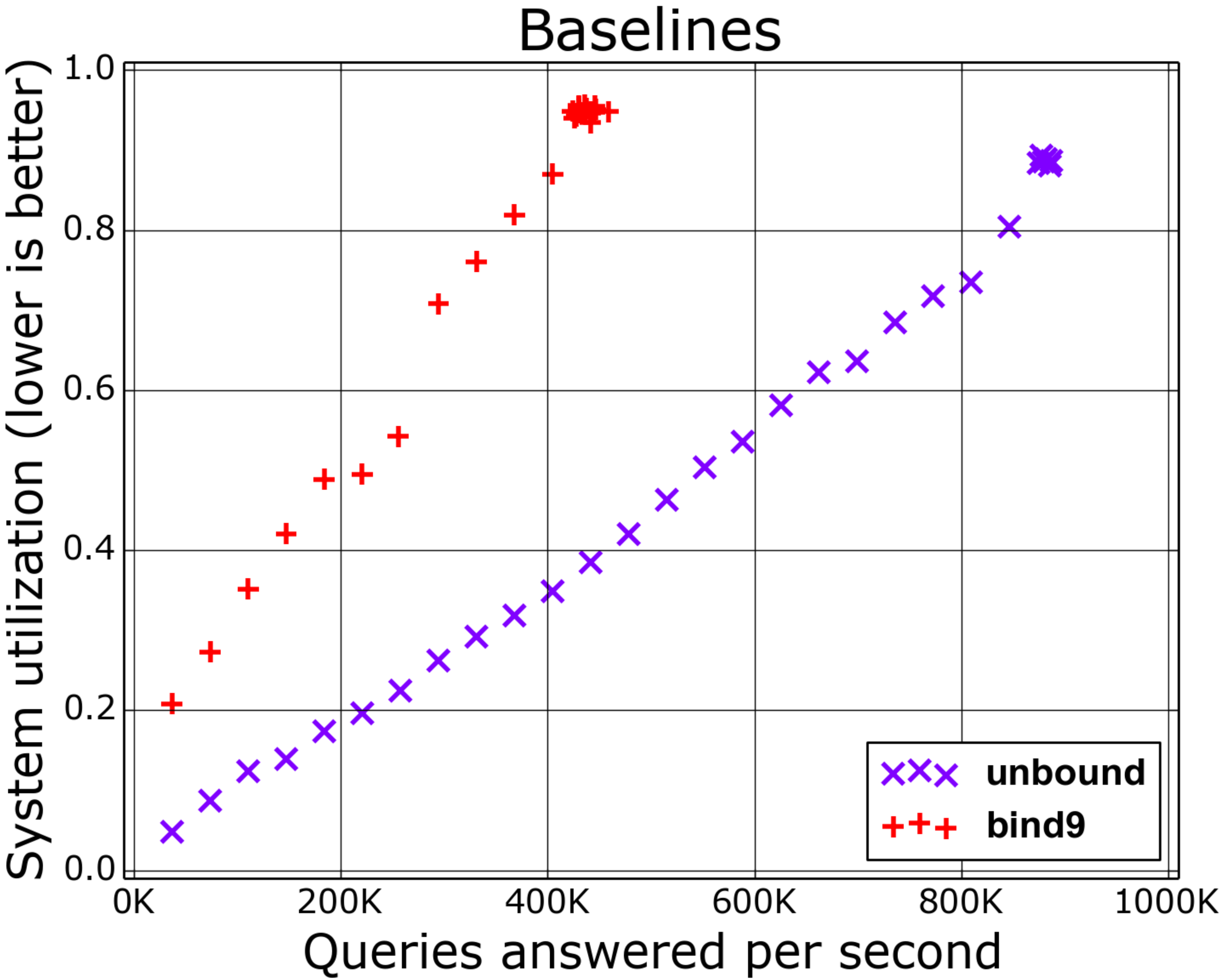
Benchmarks

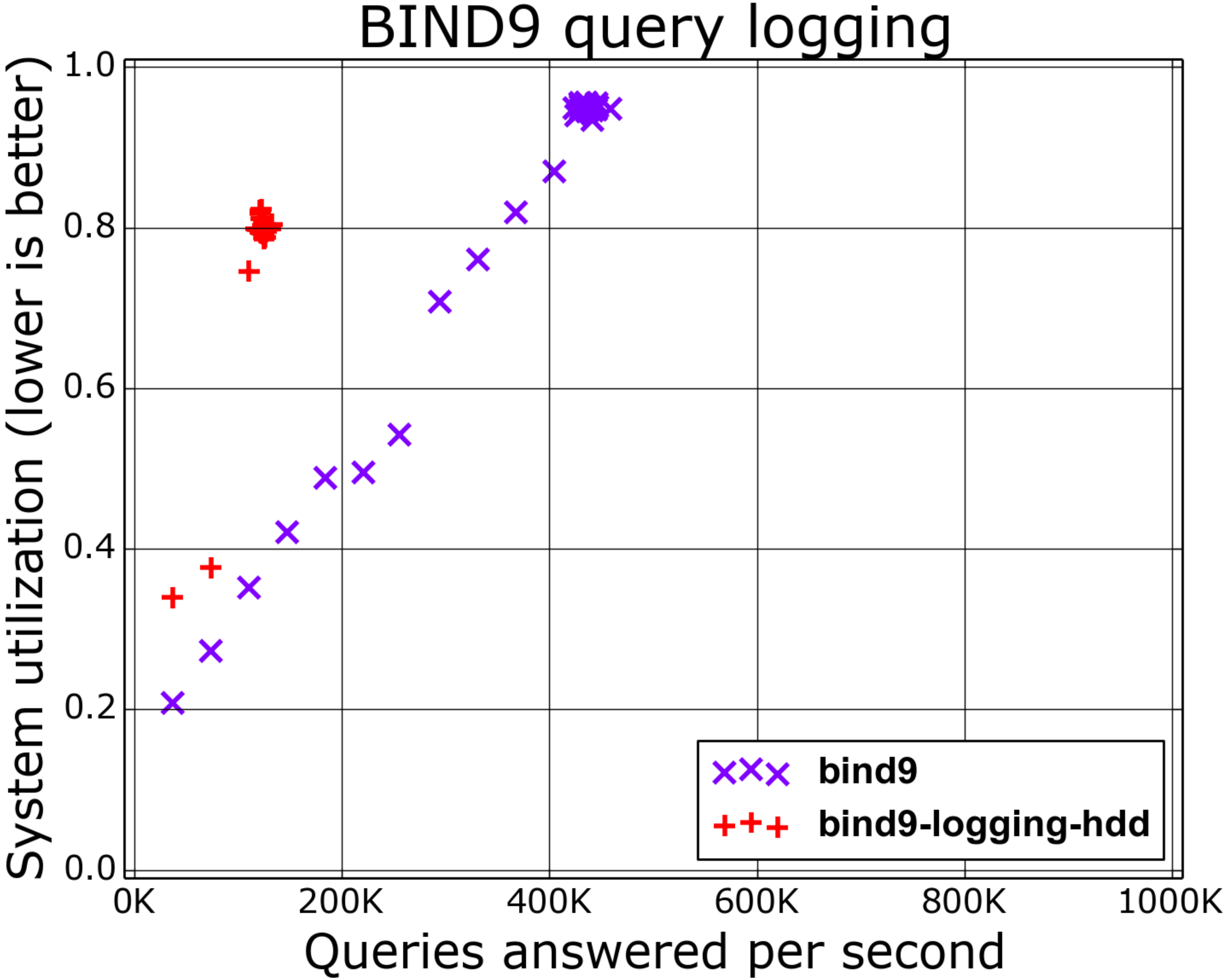
- Linux 3.11/3.12.
- Defaults, no attempt to tune networking stack.
- `trafgen` used to generate identical UDP DNS questions with random UDP ports / DNS IDs.
- `tc` token bucket filter used to precisely vary the query load offered by the sender.
- `mpstat` used to measure system load on the receiver.
- `ifpps` used to measure packet RX/TX rates on the receiver.
- `perf` used for whole-system profiling.

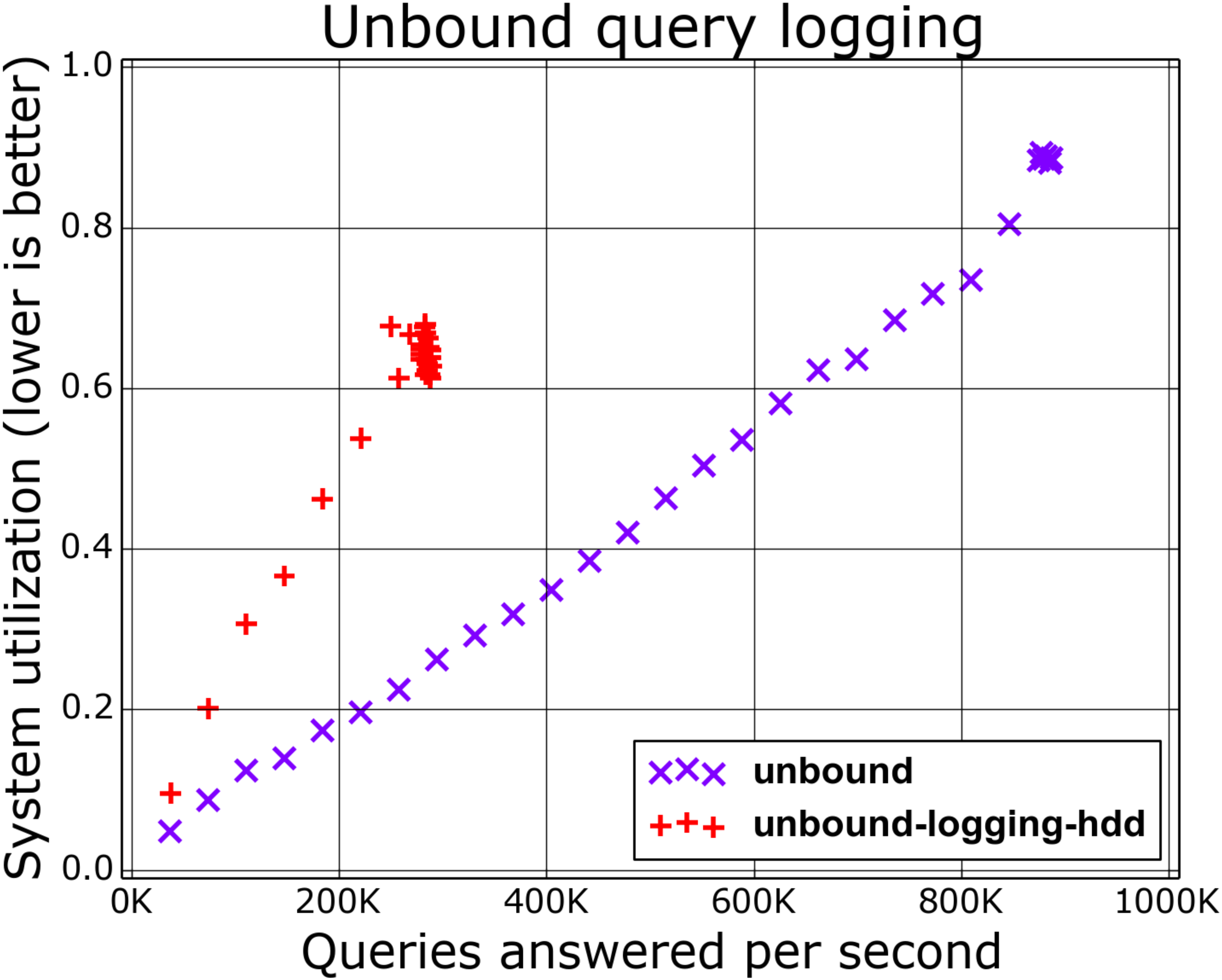
Benchmarks

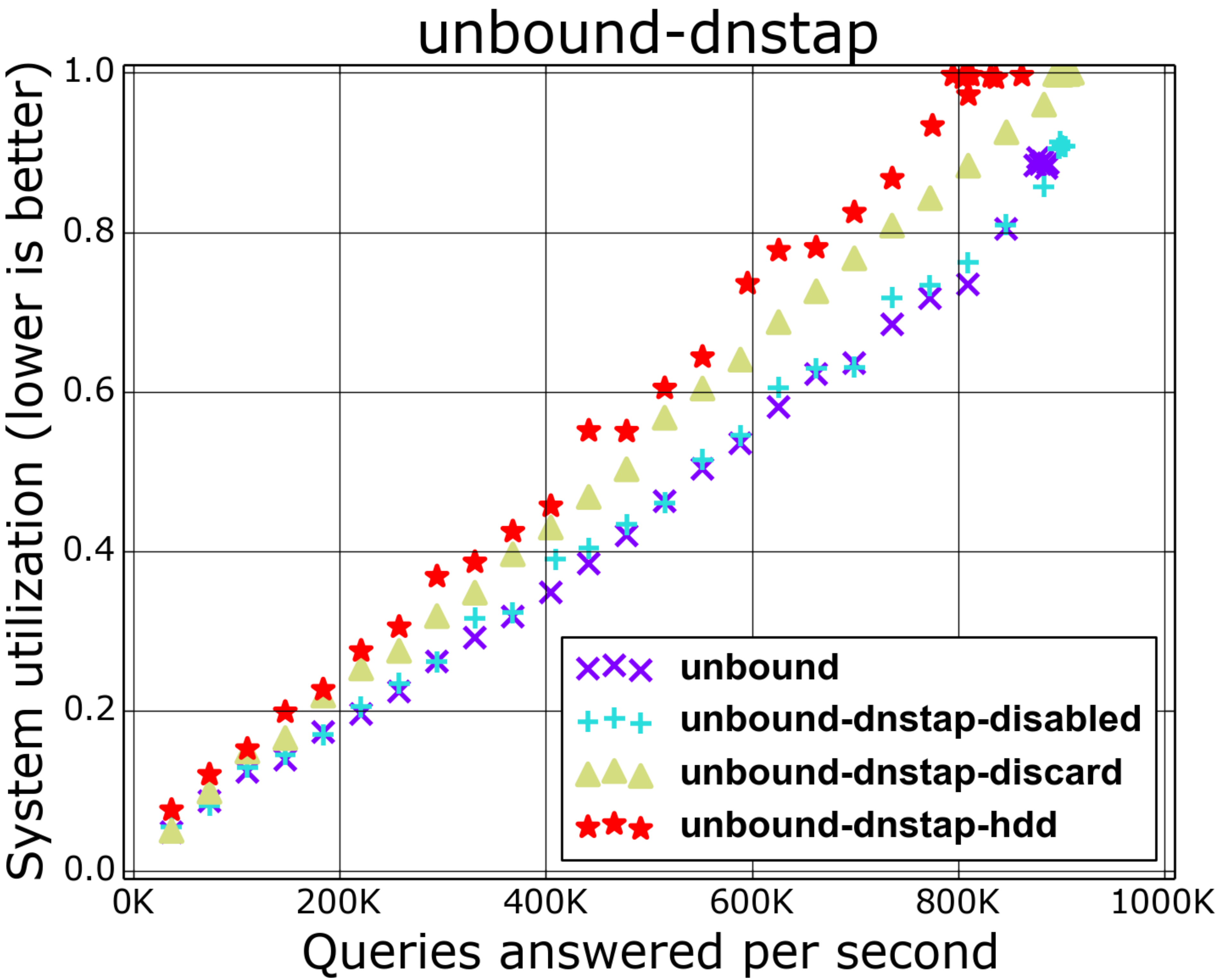
- Offer particular DNS query loads in 25 Mbps steps.
  - 25 Mbps, 50 Mbps, …, 725 Mbps, 750 Mbps.
- Measure system load and **responses/second** at the receiver, where the DNS server is running.
  - Most DNS benchmarks plot **queries/second** against response rate to characterize drop rates.
  - Plotting responses/second can still reveal bottlenecks.


Plot orientation
System utilization (lower is better)
1.0
0.8
0.6
0.4
0.2
0.0
0K
200K
400K
600K
800K
1000K
Queries answered per second
good
bad


Baselines
System utilization (lower is better)
1.0
0.8
0.6
0.4
0.2
0.0
0K
200K
400K
600K
800K
1000K
Queries answered per second
unbound
bind9


BIND9 query logging
System utilization (lower is better)
1.0
0.8
0.6
0.4
0.2
0.0
0K
200K
400K
600K
800K
1000K
Queries answered per second
bind9
bind9-logging-hdd


Unbound query logging
System utilization (lower is better)
1.0
0.8
0.6
0.4
0.2
0.0
0K
200K
400K
600K
800K
1000K
Queries answered per second
unbound
unbound-logging-hdd


unbound-dnstap
System utilization (lower is better)
1.0
0.8
0.6
0.4
0.2
0.0
0K
200K
400K
600K
800K
1000K
Queries answered per second
unbound
unbound-dnstap-disabled
unbound-dnstap-discard
unbound-dnstap-hdd

Benchmark summary

- Three recursive DNS servers were tested:
  - BIND 9.9.4, with and without query logging.
  - Unbound 1.4.21, with and without query logging.
  - Unbound with a dnstap patch logging incoming queries.

Benchmark summary

- Unbound generally scaled better than BIND 9.
- Both DNS servers implement query logging in a way that significantly impacts performance.
- dnstap added some overhead, but scaled well.

Future work

- Additional dnstap logging payload types:
  - DNS cache events: insertions, expirations, overwrites of individual resource records
- Patches to add dnstap support to more DNS software
  - Not just DNS servers!
- More documentation
- More tools that can consume dnstap formatted data
- More benchmarking
- Specifications

Summary

- Examined ***query logging*** and ***passive DNS replication.***
- Introduced new ***dnstap*** technology that can support both use cases with an in-process ***message duplication*** facility.