

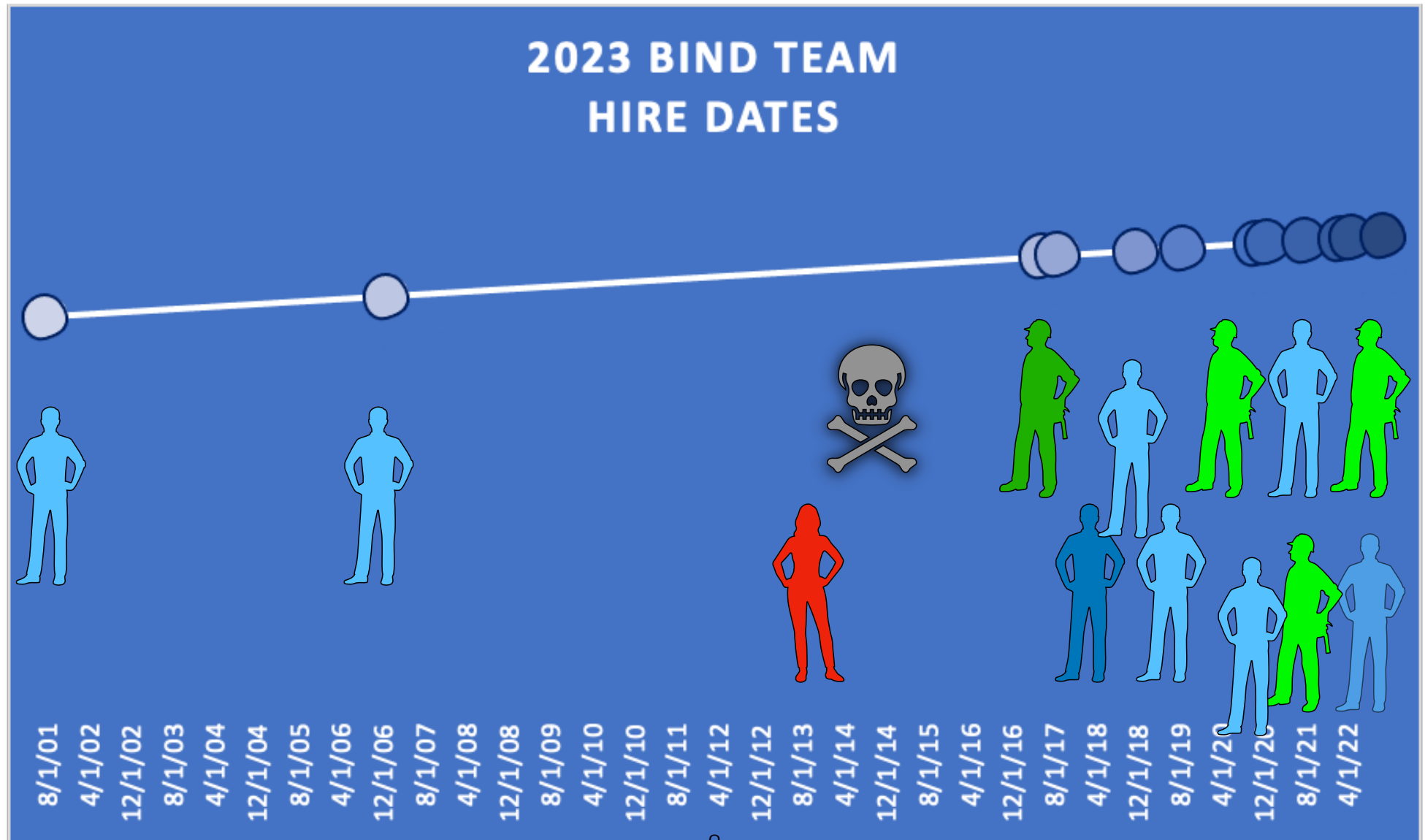
Aging Gracefully with BIND®

Vicky Risk, Product Manager
Netnod Spring Meeting, 2023



[ISC.org](https://www.isc.org)

Re-building the Team



Features added

- 2014 - 2015: BIND Cookies, Happy eyeballs, memory map
- 2016 - 2017: Catalog zones, RNDC addzone/delzone, DNSSEC Key Manager, negative trust anchors, minimal responses, DNSTAP logging, Imdb for zone list
- 2018: NSEC Aggressive Use, Serve Stale (TTL Stretching), Response Policy Zones Updates, CDS/CDNSKEY tools, ED25519 Support
- 2019: QNAME Minimization, Root zone local copy, RPZ extensions, KASP (DNSSEC key and signing maintenance)
- 2021/2022: DOT, DOH, XOT, Extended Errors

Not all features are improvements

- Serve stale - cache management was *already complicated* but now
- NSEC aggressive use - sounds good, but it can be faster to just send a query
- QNAME minimization - privacy is a good goal, but this caused some subtle collateral issues

Refactoring

- 2018/2019: Refactored RPZ, query_find(), answer_response()
- 2019: C99 Support in Compiler, POSIX Threads, Advanced Sockets API for IPv6, Standard Atomic <stdatomic.h>, Support for lot of old systems dropped (Windows!)
- 2020: Crypto refactoring, OpenSSL is now mandatory, Task Manager is now multithreaded, Socket Code has multiple event loops
- 2021/22: Interface manager refactored, libuv replaced ISC code
- 2022: Dispatch manager refactored, jemalloc replaced ISC's homegrown memory allocator, label compression, began qp-trie experiments
- 2023: new database (targets: zone list, zone contents, forwarding table, cache), dns stream (DOT & DOH)

Why refactor?

- Code Complexity
- Leverage modern HW & OS
- Adopt standard LINUX libraries
- Deprecate obsolete features

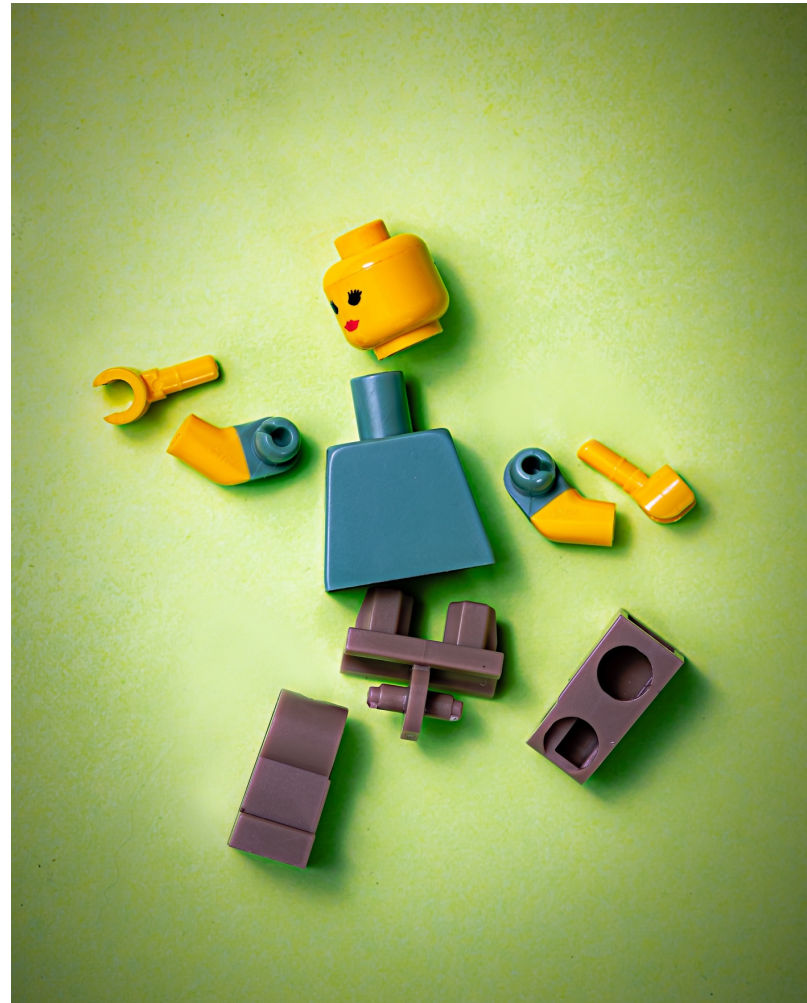


Photo by [Jackson Simmer](#) on [Unsplash](#)

Complexity*

Recommended

BIND 9.10

<10 Easy to Maintain

535 functions over 20

11 – 20 Harder to maintain

93 functions over 50

21+ Candidate for
refactoring/redesign

26 functions over 100

<https://www.isc.org/blogs/bind-9-refactoring/>

Evidence - query_find()

Before: complexity factor of 453

After: *worst* function had complexity 50

<https://www.isc.org/blogs/bind-9-refactoring/>

Computing Changed

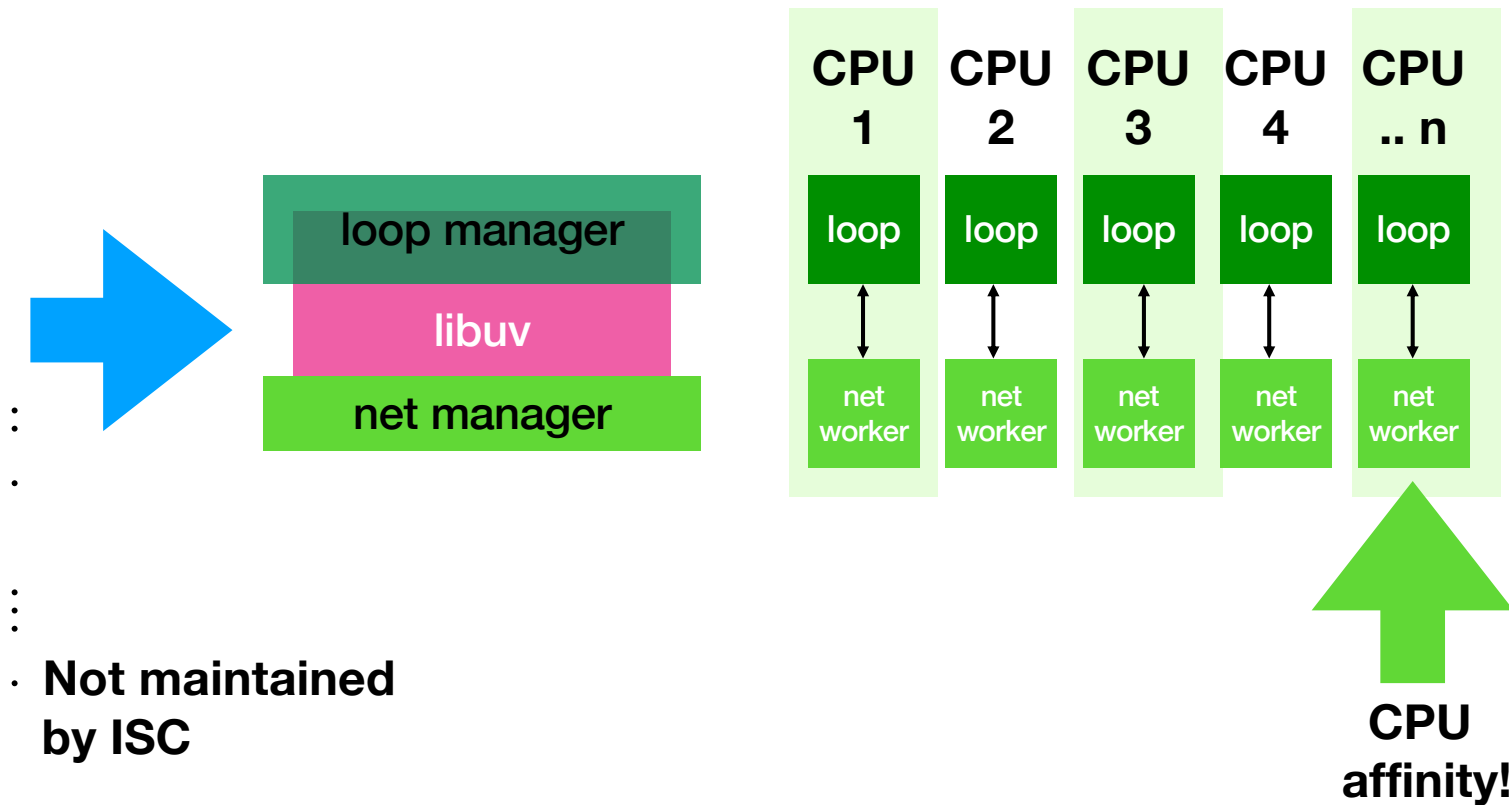
1998

- Quad-CPU hw, 500 Mhz, 100 MB network
- AIX, HP/UX, Solaris, OS/2, NeXT, Windows NT, Ultrix, VMS.

2023

- 256 CPU-threads, 2.4 GHz, 10/100 Gbps network
- Linux. BSD.

New Network Interface



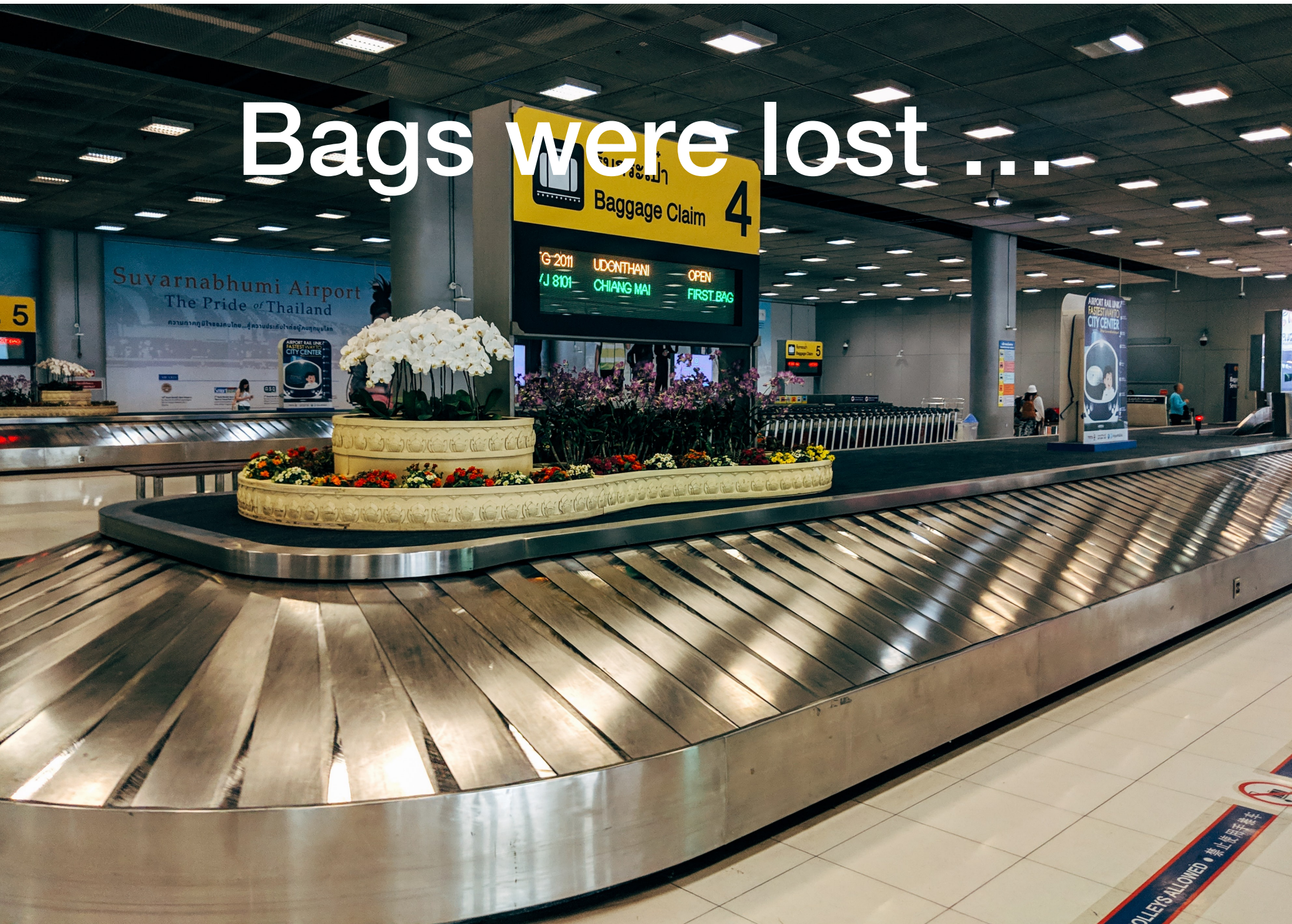
Single listener



Multiple listeners



Bags were lost



Network Refactoring in 9.16 was Unfinished

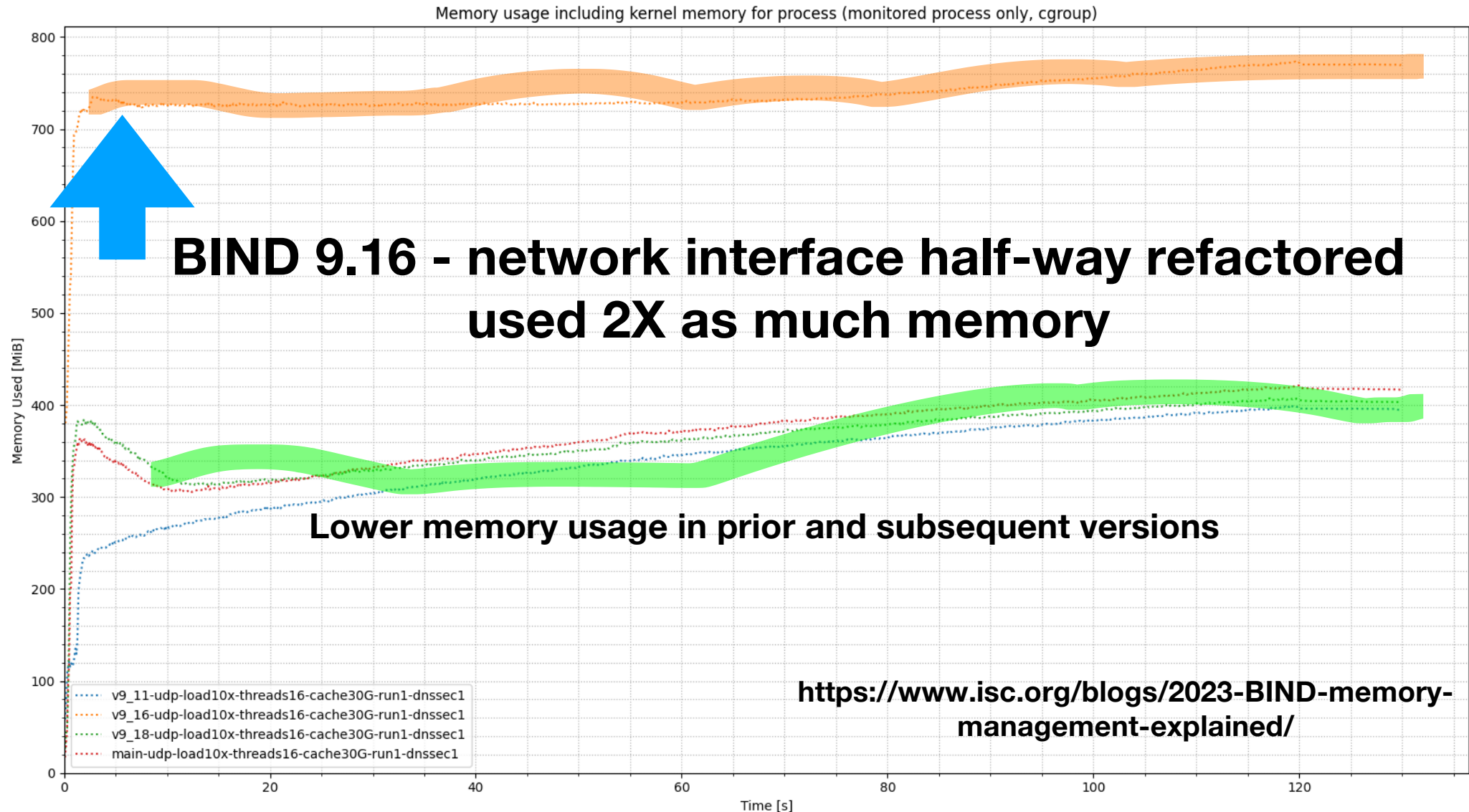


New Multi-listener model, with libuv, etc was used to
receive requests to named from some other service.



Old BIND task manager and home-grown socket code was used
for when named wanted to send something **out** and listen back for a response
(recursive query, dig, notify, zone XFR request)

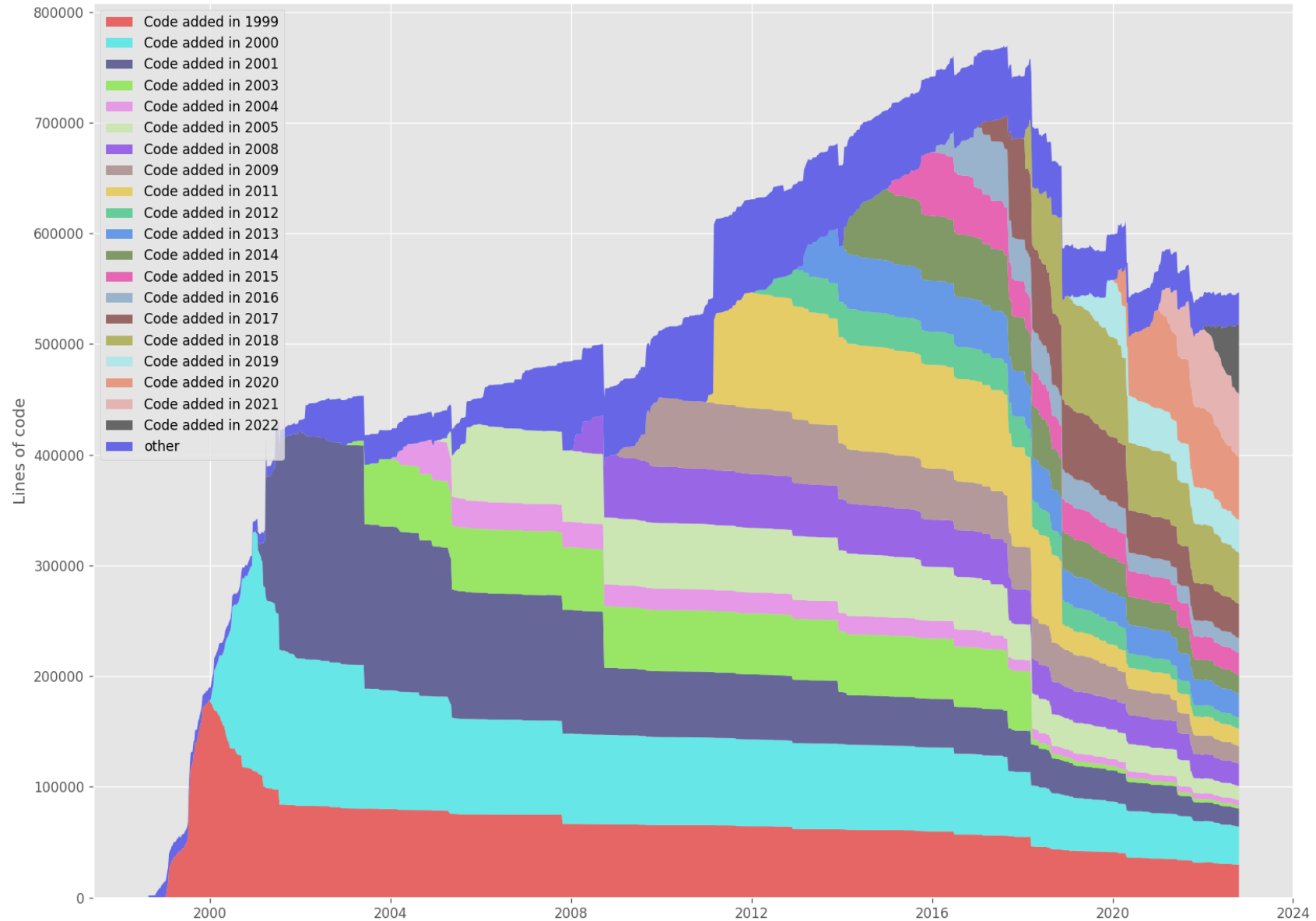
Memory Usage



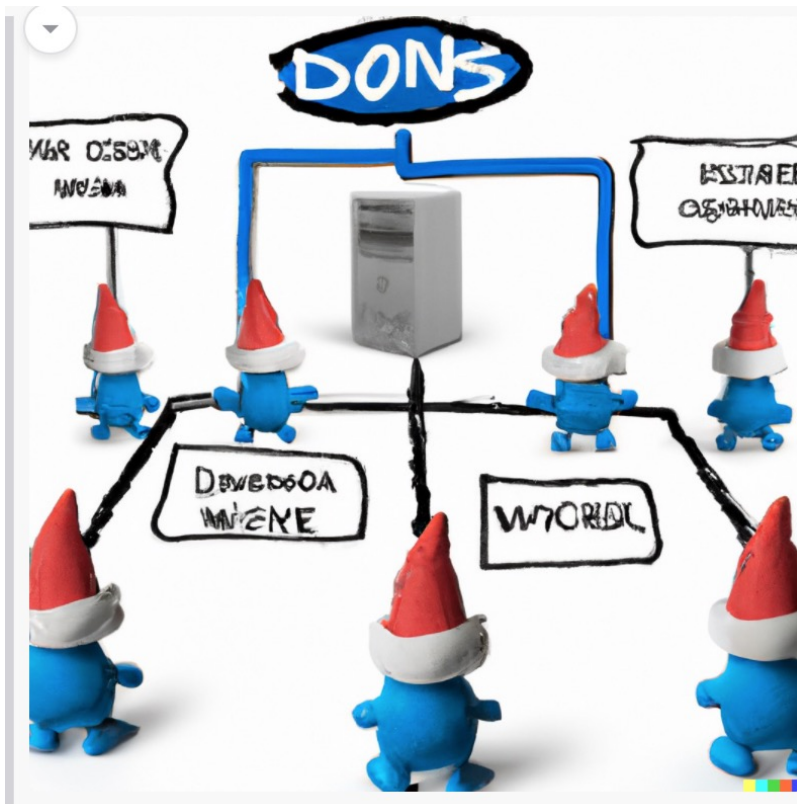
Is Refactoring Progress?

- What if the developers are just messing around?
- All change has risk
- Where is the user-benefit?
- How much of the code has been re-factored so far, what else needs to be done?

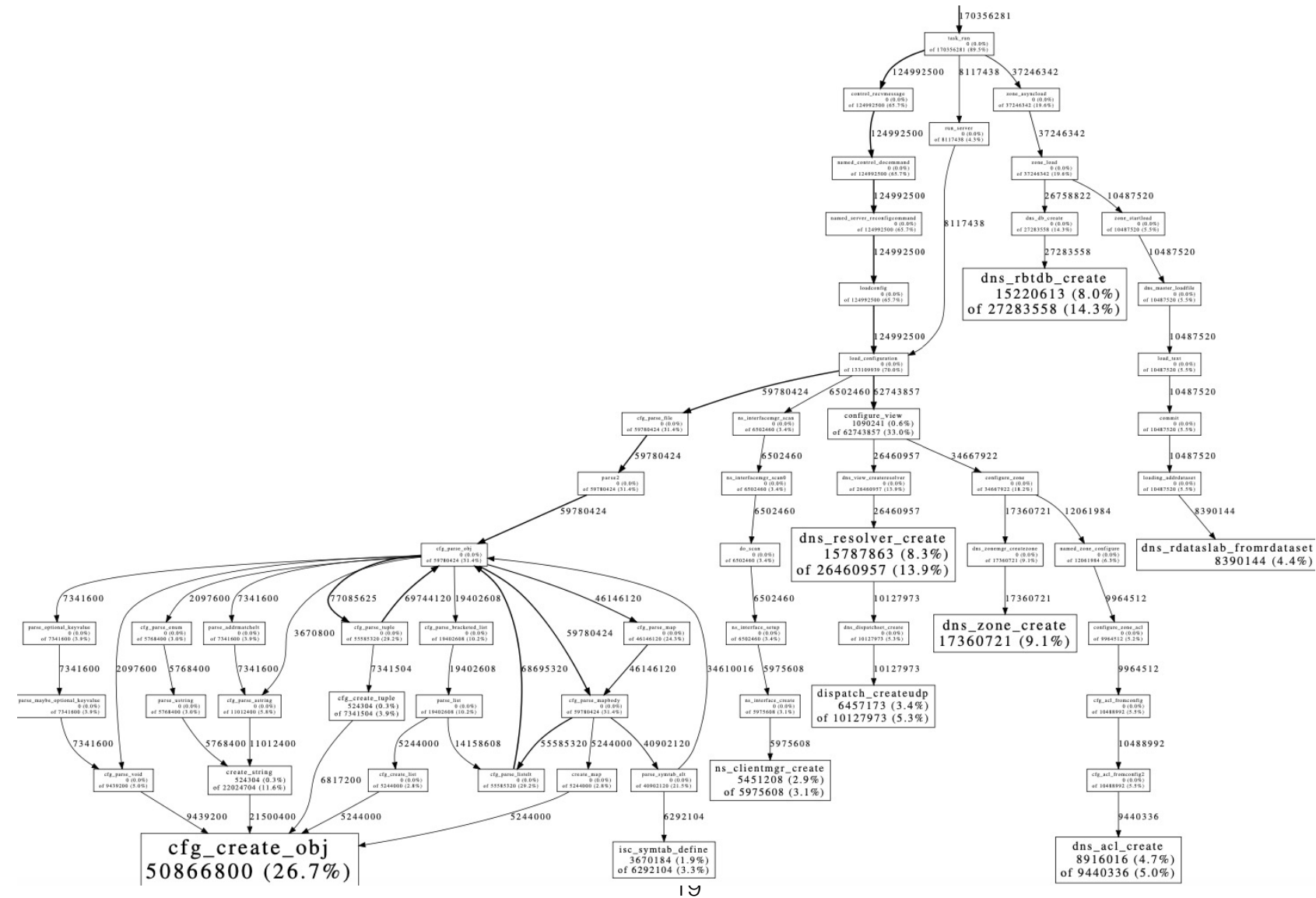
How old is this code?

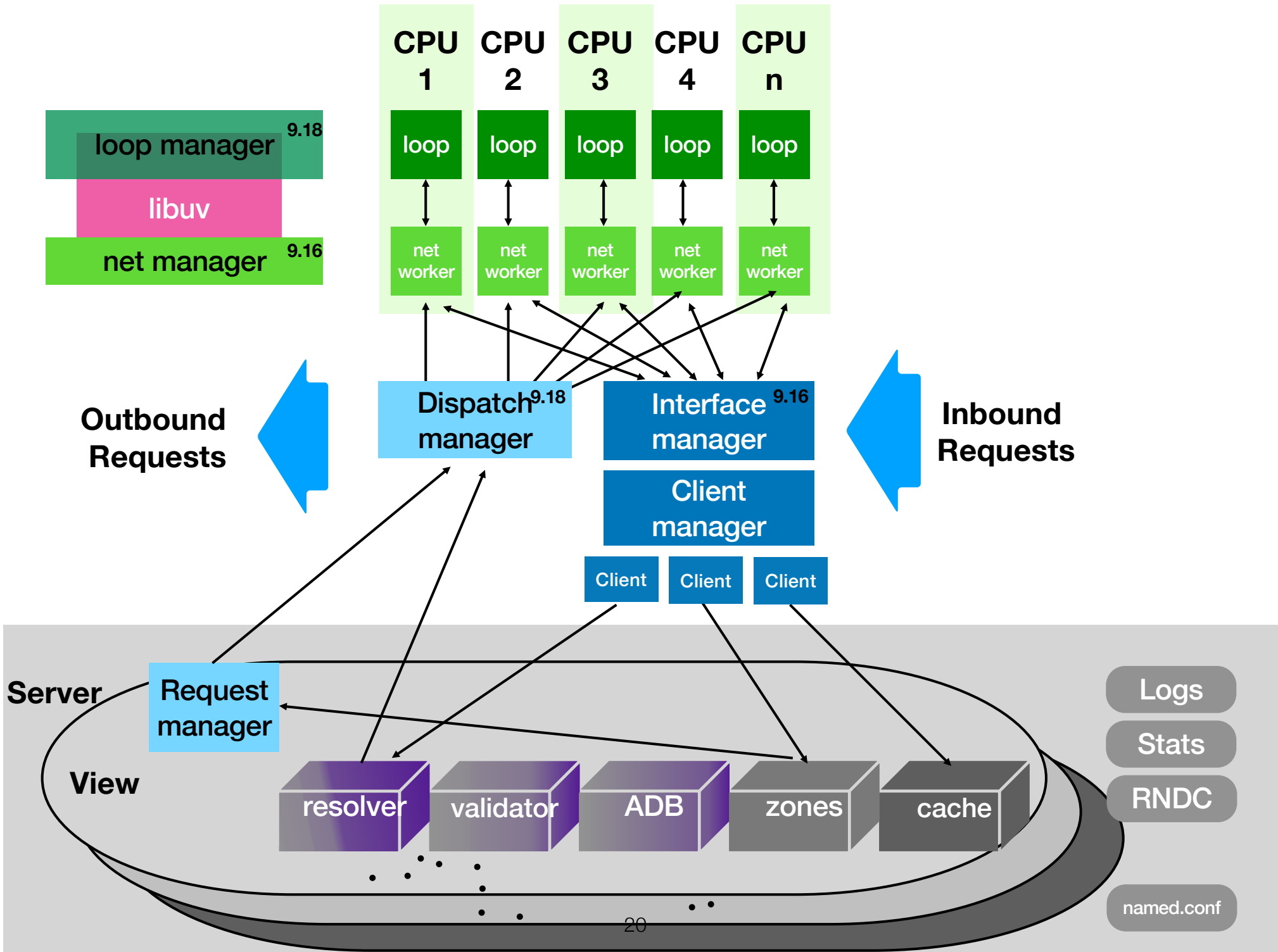


Attempts to get a BIND functional diagram



Or.. way too much detail





How much is refactored?

Refactored
~177KLOC

- Client manager & interface manager: 23KLOC
- Resolver, validator & ADB: 22KLOC (resolver has been refactored, but not validator or ADB)
- LIBDNS (includes Dispatch mgr, zones, databases, DLZ, OpenSSL wrapper: 168KLOC) - (zones, databases and DLZ have not been refactored ~36KLOC)= 132KLOC

Not Refactored
~321KLOC

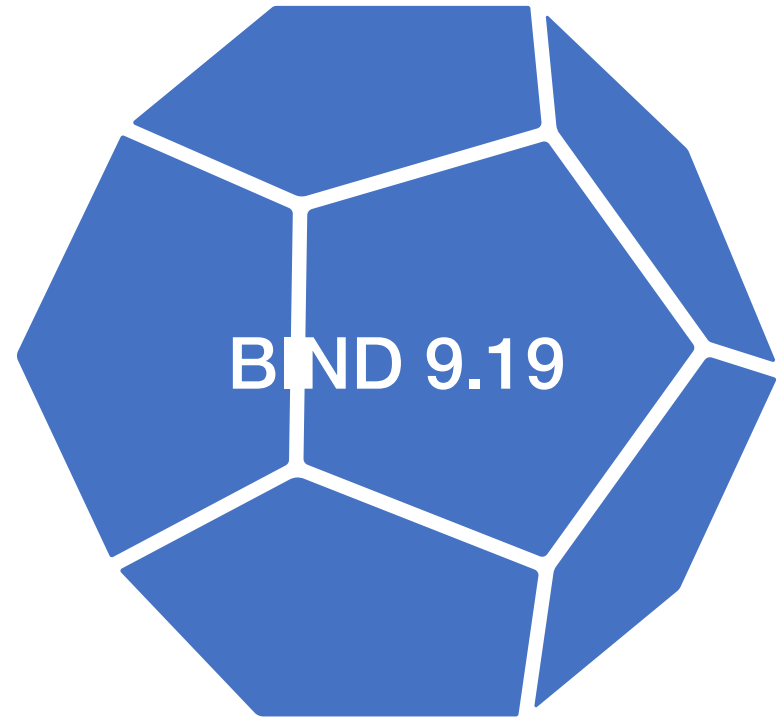
- View: 2.5KLOC
- Zones: 23KLOC + 13KLOC for database (this is the next target for further refactoring)
- Everything else: 283 KLOC (named.conf parser, random number generator, logging, statistics, RNDC...)

LESS code



2014

954 c files - 549,858 LOC

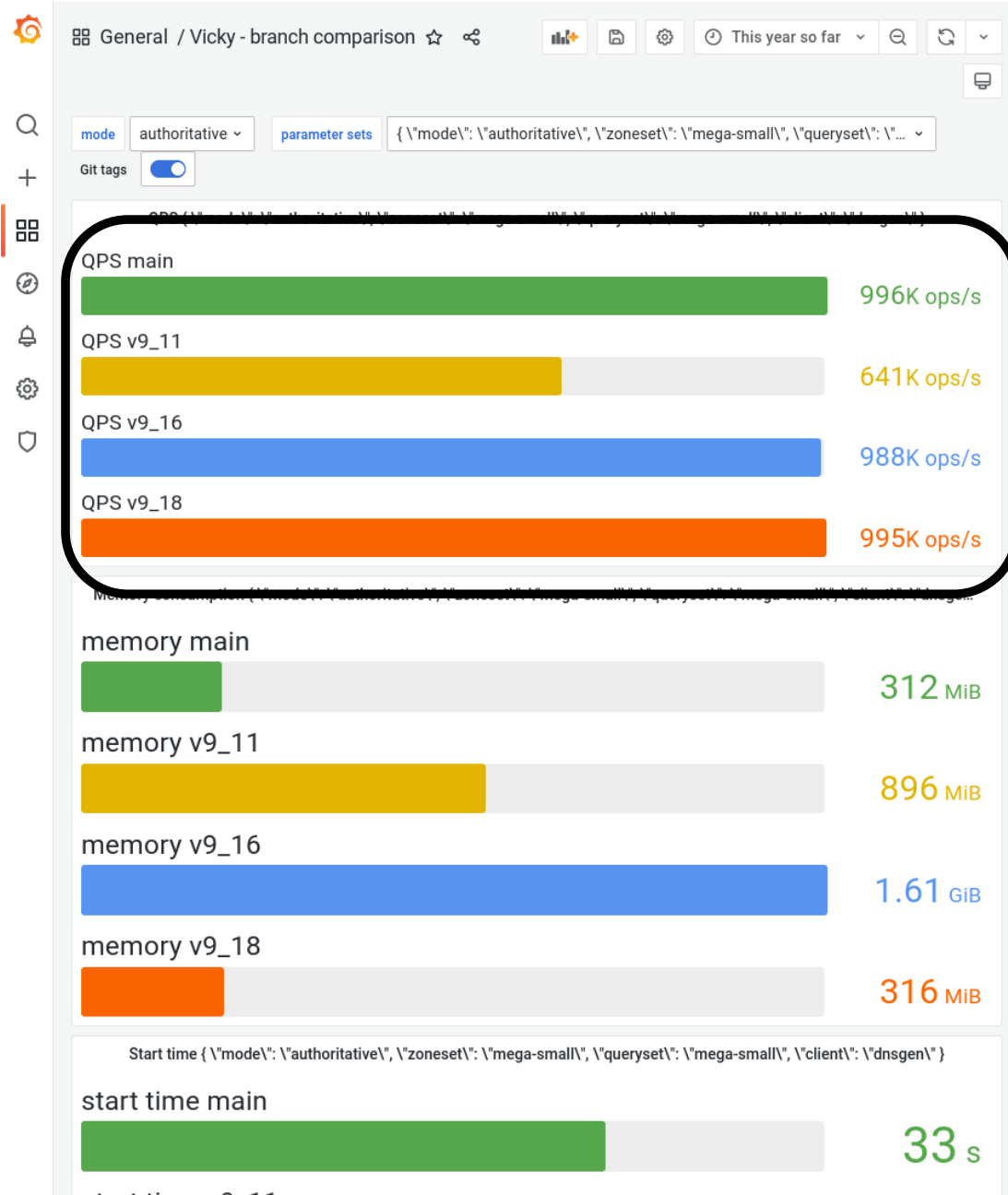


2023

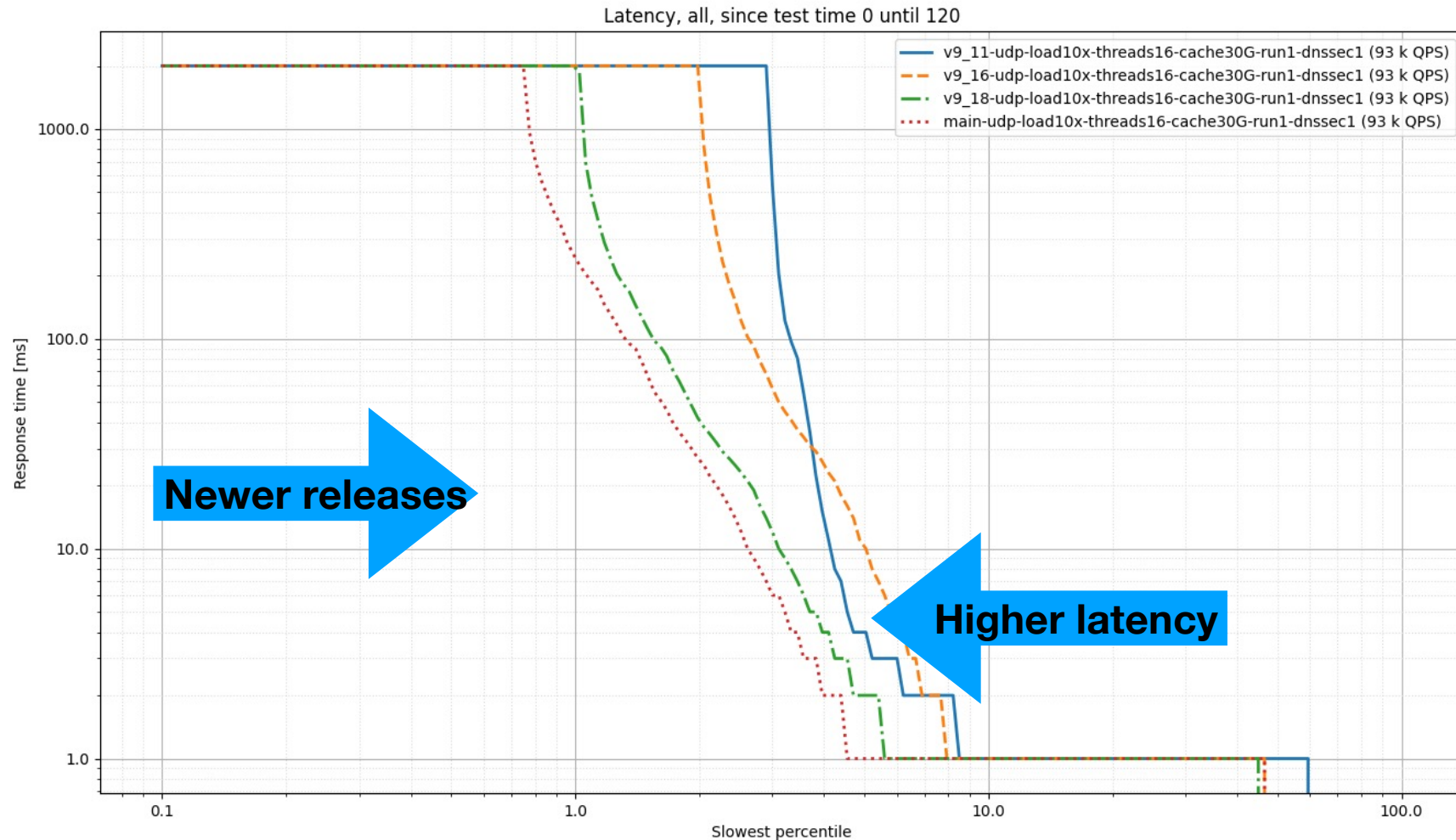
727 c files - 474,249 LOC

Authoritative Performance Improved Dramatically 9.11 -> 9.16

<https://www.isc.org/blogs/bind-performance-march-2020/>



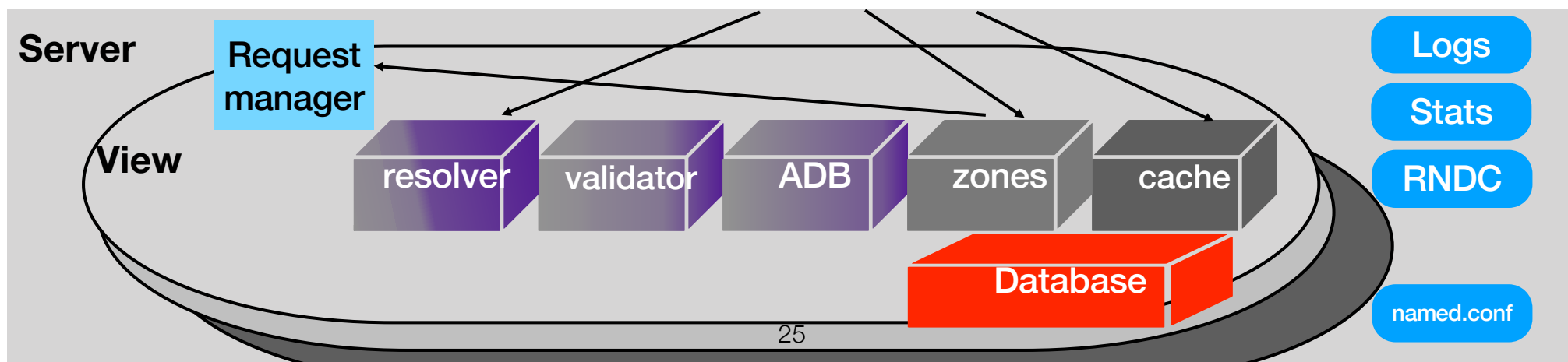
Resolver – lower latency



<https://www.isc.org/blogs/bind-resolver-performance-july-2021/>

What's next

- Qp-trie will replace (possibly as an option) the RBTDB, in BIND 9.20
- Will eventually support the zone list, zone contents, and the cache
- This may improve performance, but the primary goal is to remove conflicts and blocking (e.g. zone transfers interfering with query responses)



Project Improvements

- Open repo, open issue tracker
- ISC-maintained packages for CentOS/Oracle, Debian, Ubuntu, Docker
- Regular monthly development & maintenance releases
- Bi-annual stable versions, each supported for 4 years
- Docs on-line at bind9.readthedocs.io, new hyperlinks, topics, more how-to and intro content

Quality Assurance

- Unit tests + system tests
- CI: about 70 jobs are run for every revision of each merge request
- *All the code sanitizers and static analyzers* (ASAN, UBSAN, TSAN). Both GCC and Clang. Coccinelle, PyLint, flake8, Danger.
- Daily test runs: **performance tests**, resp-diff
- Fuzzing
- 2-week process post-code freeze for analyzing test results, investigating failures, upgrading OSes, doc reviews: checklist-driven.

CVEs

- Yes, we still have CVEs (but now we find more of them ourselves!)
- We have raised the bar for the full security response to $CVE \geq 7.0$
- We still don't have a good defensible metric for ***risk*** (as opposed to severity)

Conclusion

the dev and QA teams added new, energetic talent

entire project is much more transparent

added features to make DNS operations more robust vs
DDOS, abuse

simplified maintenance, DNSSEC signatures, improved docs

modernizing the infrastructure is improving performance &
allows us to add new transports

References

- Ondrej's Memory Management blog: <https://www.isc.org/blogs/2023-BIND-memory-management-explained/>
- Tony Finch's blog on qp-trie: <https://dotat.at/@/2023-02-28-qp-bind.html>
- Some QA blogs: <https://www.isc.org/blogs/jemalloc-glitch/>, https://www.isc.org/docs/Detecting_latency_spikes_in_DNS_server_implementation.pdf, <https://www.isc.org/blogs/a-curious-case-of-intermittent-bind-system-test-failures/>
- BIND9 2022 Retrospective: <https://www.isc.org/blogs/2022-bind9-retrospective/>

IF you are running ISC DHCP



kea.isc.org