NSD
An Authoritative Nameserver

A peek under the hood of version 3

Olaf Kolkman

olaf@nlnetlabs.nl

Outline

- Background on NSD: what, when, who
- Design and Architecture: goals and discription
- NSD3
- DISTEL: Regression and Performance

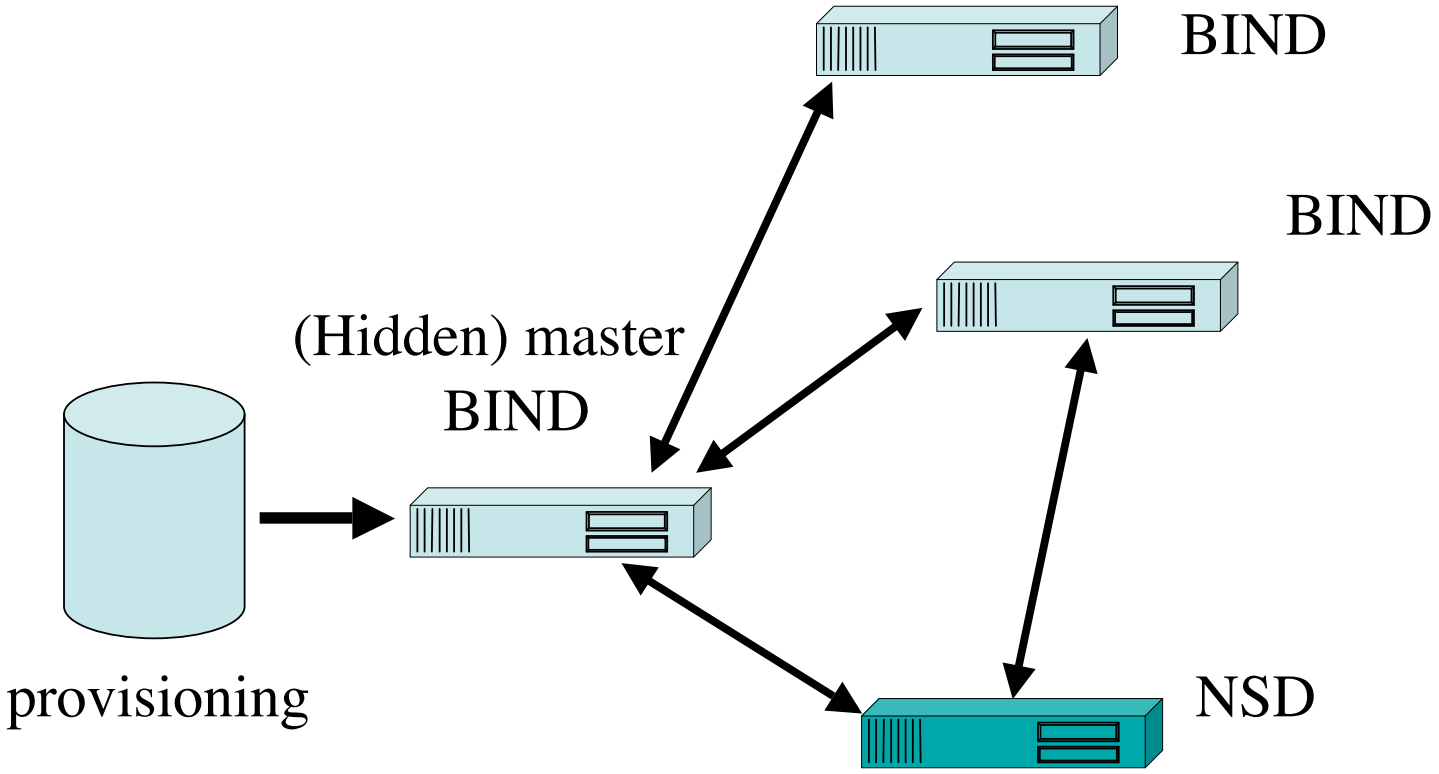
What Is NSD

- NSD is an authoritative only nameserver
  - High performance
  - Lean and mean
  - RFC compliant
- NSD is developed and maintained by NLnet Labs
  - Not for profit "Open Source Lab"
  - In house DNS expertise

NSD history

- Conceived in 2000
  - Convergence seen on root and TLD level towards one implementation (BIND)
  - inbreed increases the thread of eradication
  - Biological diversity improves the stability of a species
  - Inspiration and Development in close cooperation with RIPE NCC
- Independent reference implementation with specific design goals

Typical NSD Use Case



NSD users

- Used on root servers
  - k.root-servers.net, h.root-servers.net
- 19 out of the 885 TLD servers use NSD
  - According to fpdns
  - Include TLD servers for .NL, SE, AT, DK, CZ

Outline

- Background on NSD: what, when, who
- Design and Architecture: goals and discription
- NSD3
- DISTEL: Regression and Performance

Design Goals

- Conformity to the relevant DNS RFCs
  - Document interpretation in case of ambiguity
- Code diversity from other implementations
  - Written from scratch
- Authoritative server only
- Regression tested against bind8/9
  - Understanding differences
- Resilience to high load
  - To cope with DOS

More Design Goals

- Open source
  - From first public release
- Documentation
  - Operation and inside code
- Reviewed code
  - Internal review and tests
- Simplicity
  - Simple == Secure
- Reasonable Portability
  - Modern *NIX Oss (FreeBSD, Linux, Solaris, OS X etc)

Explicit non-goals

- No caching
  - Not even to optimize for fast responses
- No slavish responsiveness
  - Be able to adapt to DOS
- No end-user "friendliness"
  - Not cuddling users with GUIs
  - Assume knowledge of the OS and of DNS
- No creeping featurism
  - Such as random order RR in RR set

Important features through time

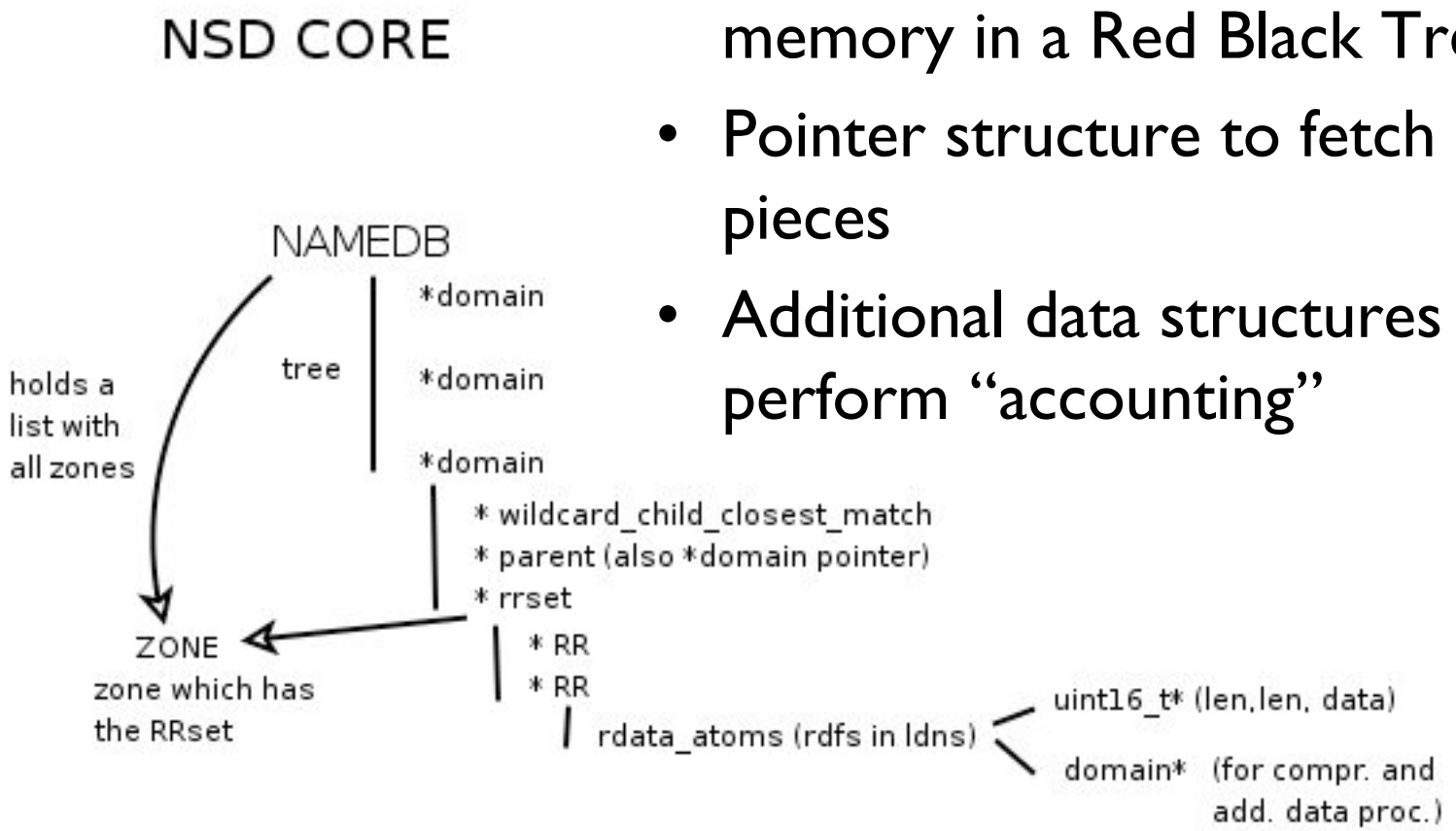
- NSD 1.0.0 as a master server only
- NSD 1.0.1 support for AXFR through external tools
- NSD 2.0.0 Support for DNSSEC bis
  - Internal DB design change
- NSD 3
  - Support for IXFR
  - Improved IPC
  - DNAME support

NSD Architecture Main features

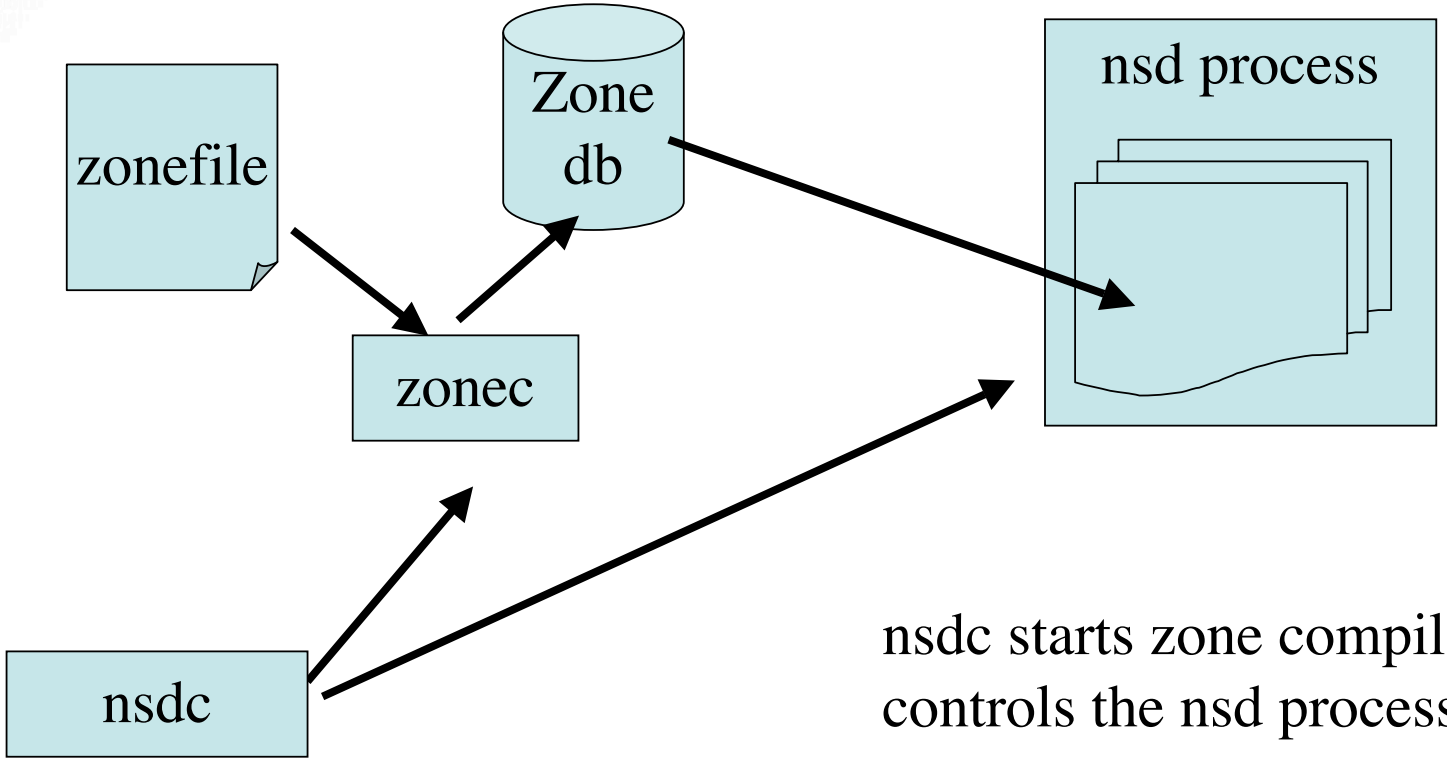
- Pre-compile answers as much as possible and perform as little work as possible during serving
  - NSD1 had fully compiled answers
    - Only some name compression at run-time
  - NSD 2 only compiled RR sets
    - Assembly at run-time
    - Mainly to enable support of DNSSEC
    - Small performance penalty

NSD Data

- Precompiled data stored in memory in a Red Black Tree
- Pointer structure to fetch all pieces
- Additional data structures to perform “accounting”

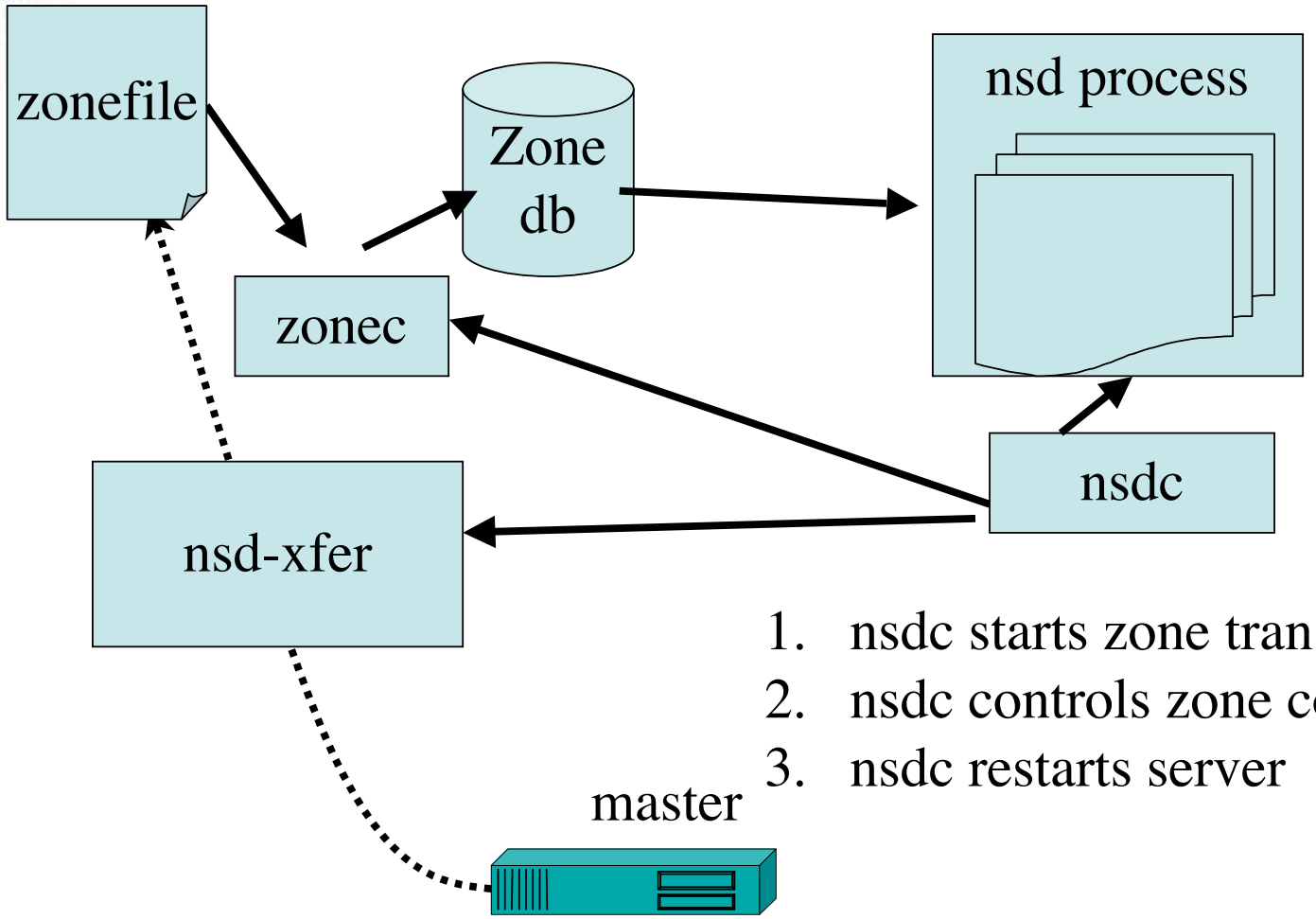


NSD1/2 operation model (zone loading)



nsdc starts zone compilation and controls the nsd process

NSD1/2 operation model
Zone transfer



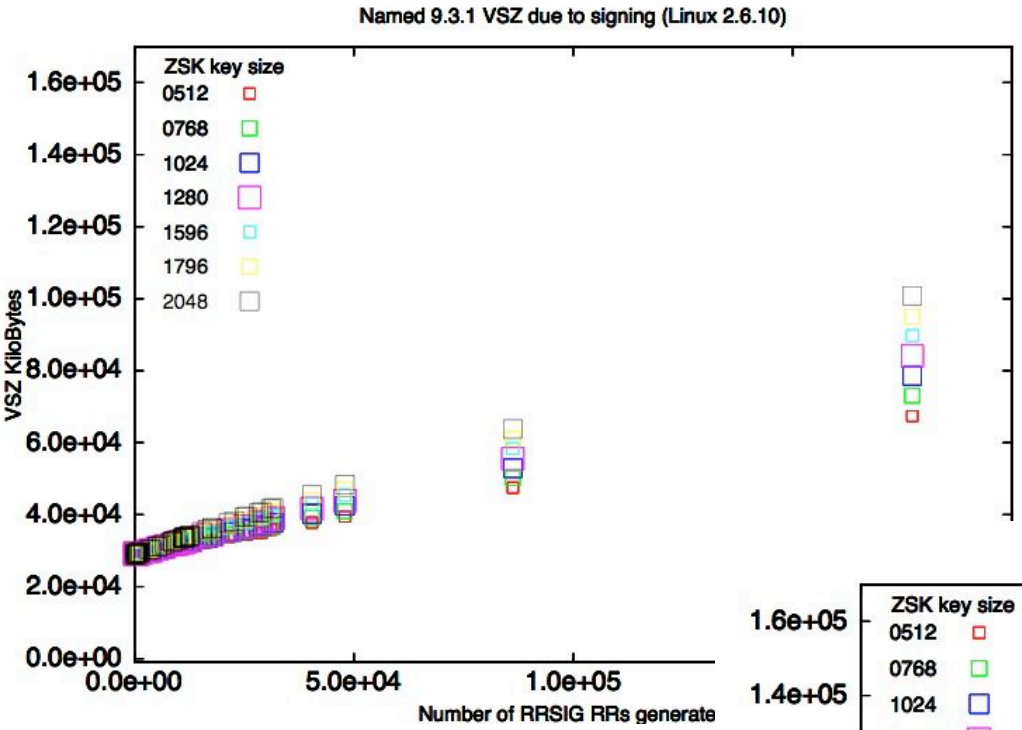
1. nsdc starts zone transfer process
2. nsdc controls zone compilation
3. nsdc restarts server

NSD 2 Operational Features

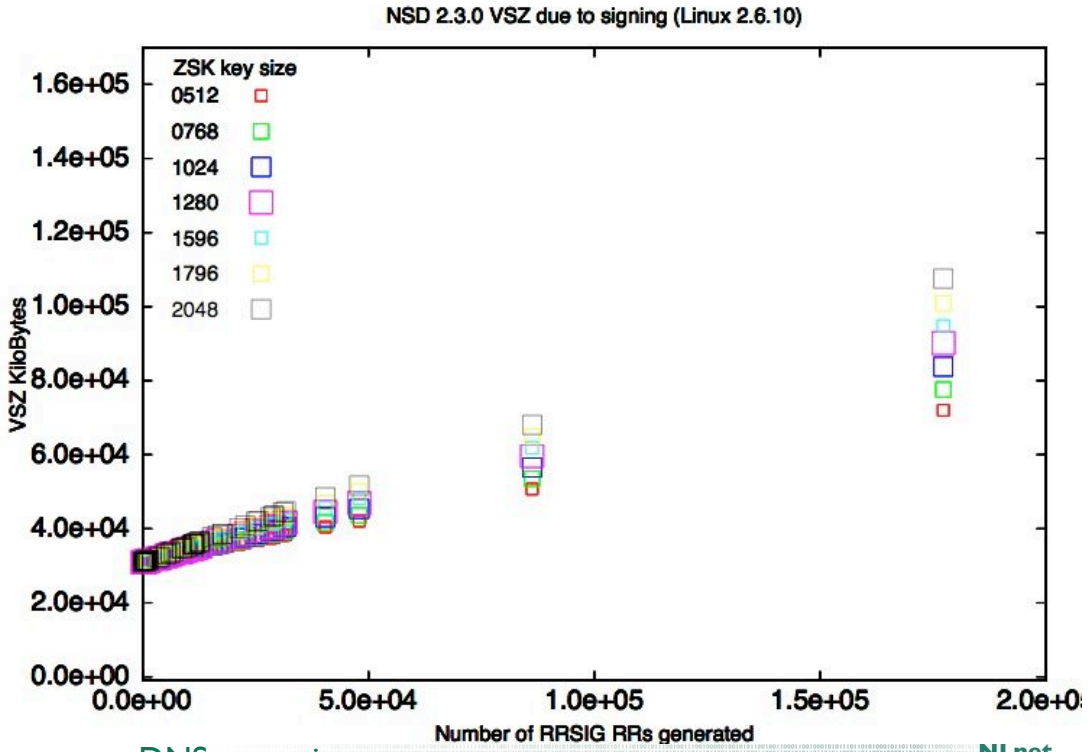
- Requires ‘cron’ and/or manual control for ingress zone transfers
- .NL zone signed with 1024big ZSK

	unsigned	signed	
DB file	46	251	MB
Core	109	388	MB

- Memory characteristics for DNSSEC similar to BIND (graphs next slide)



vsz as function of the number of RR sigs generated for various key sizes



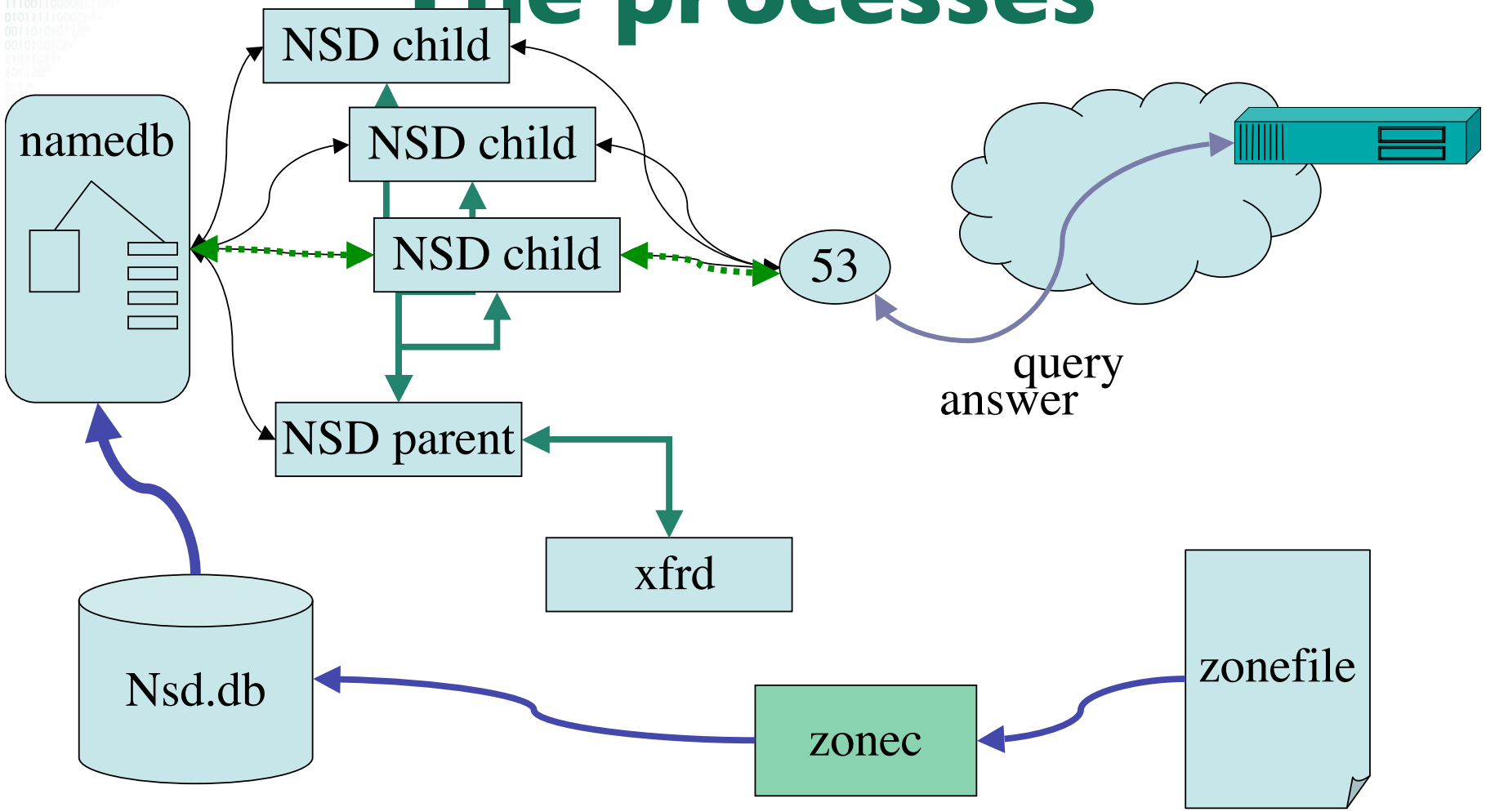
Outline

- Background on NSD: what, when, who
- Design and Architecture: goals and discription
- NSD3
- DISTEL: Regression and Performance

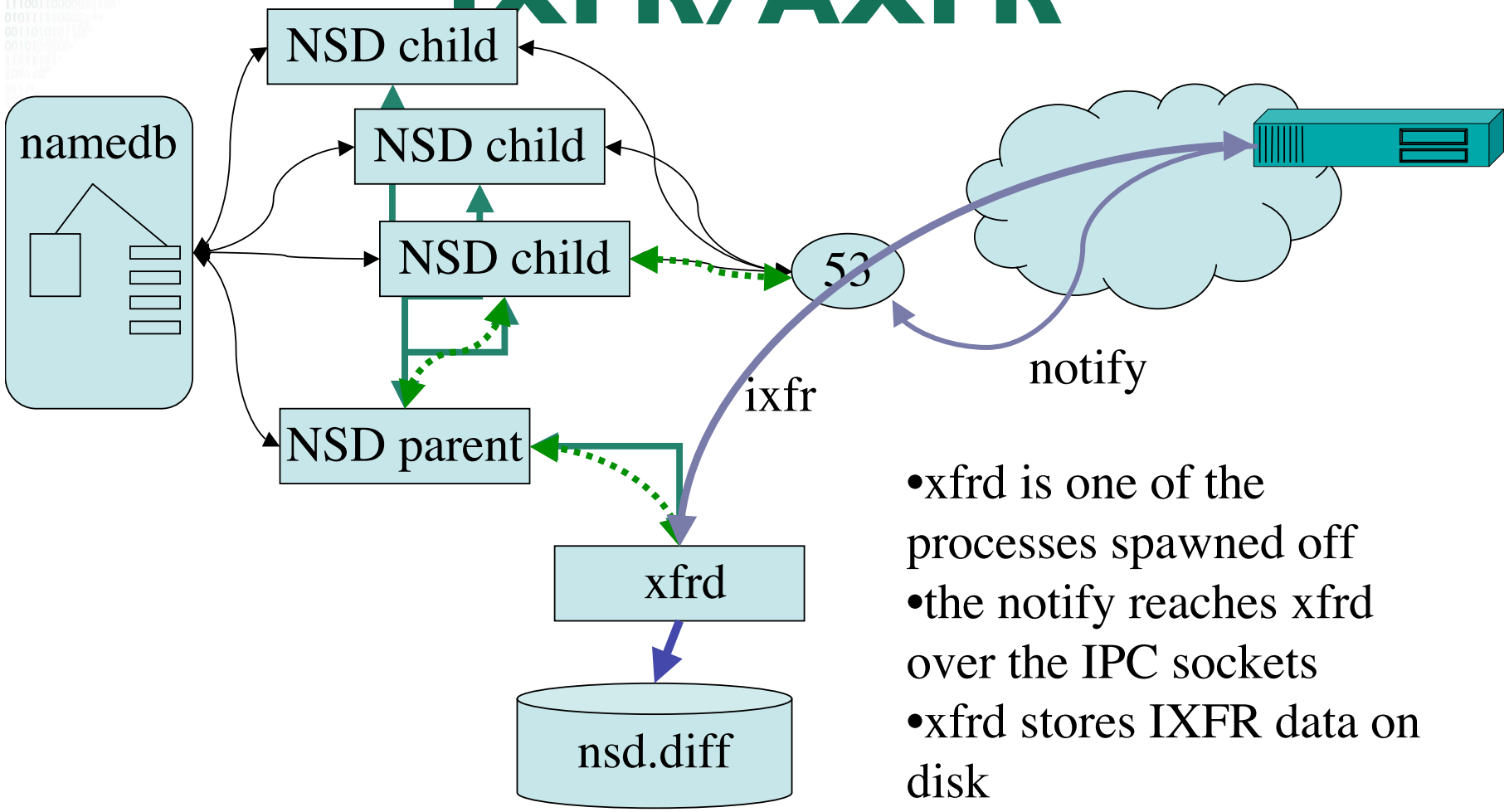
NSD 3
Wish list

- Incremental update support
  - Full zone network transport and recompilation is expensive
- Cronjob triggered AXFR does not really support SOA timings
- DNAME support
  - Recent ICANN announcement w.r.t. testing IDN support in the root
- NSEC3 support

NSD3 Architecture
The processes

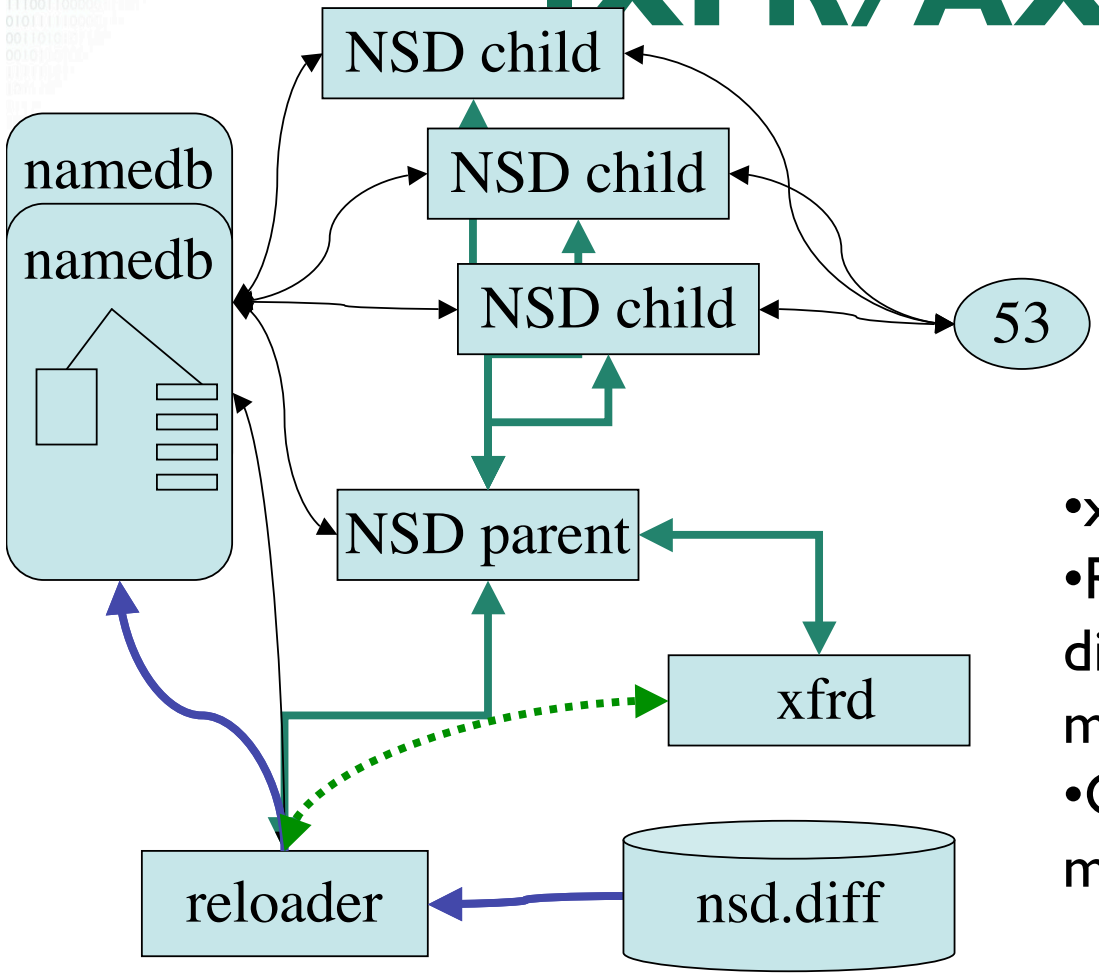


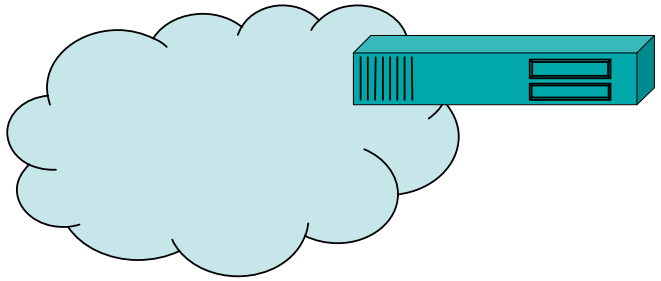
NSD3 Architecture IXFR/AXFR



- xfrd is one of the processes spawned off
- the notify reaches xfrd over the IPC sockets
- xfrd stores IXFR data on disk

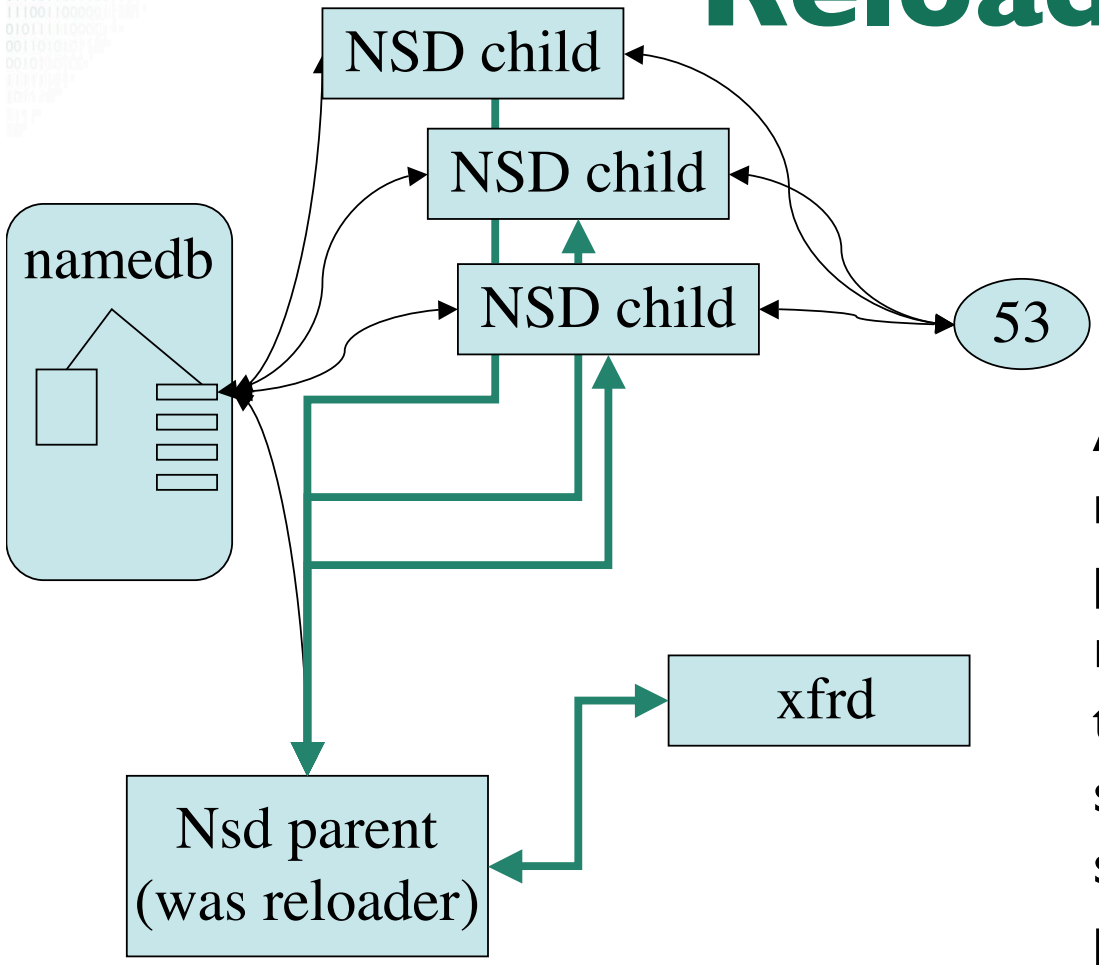
NSD3 Architecture IXFR/AXFR

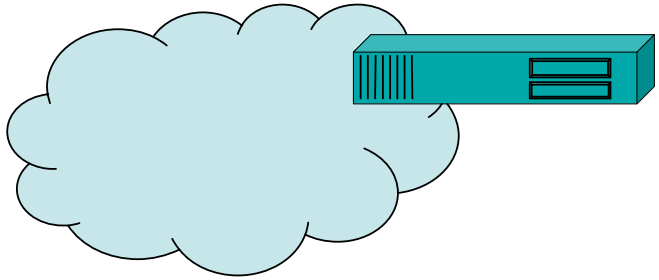




- xfrd triggers the reloader
- Reloader merges the differences into the memory
- Copy-on-write minimizes memory overhead

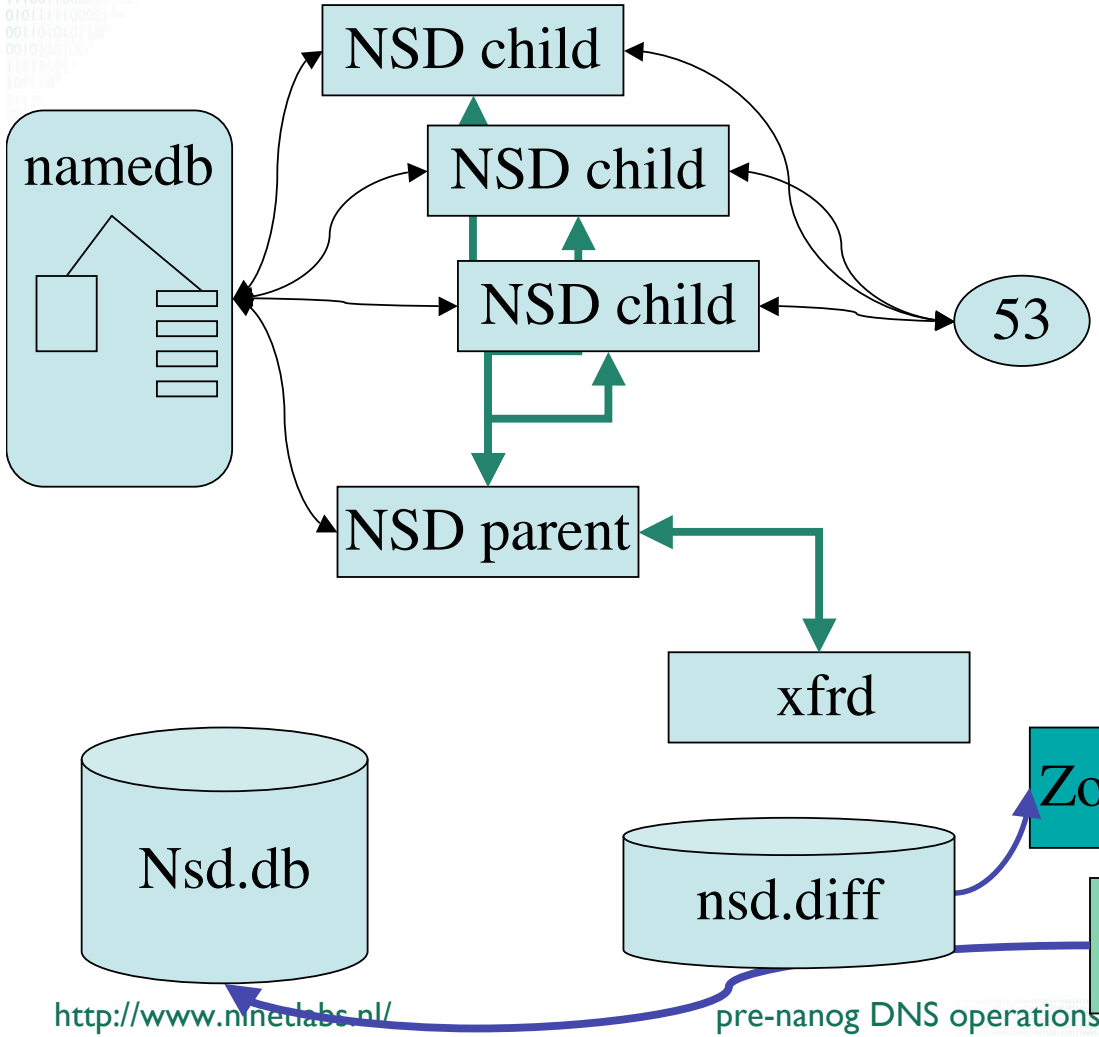
NSD3 Architecture Reload





After updating the memory the reload process kills all children, restarts new children that can access the shared memory and syncs state with the xfrd process

NSD3 Architecture
Cleaning diff files



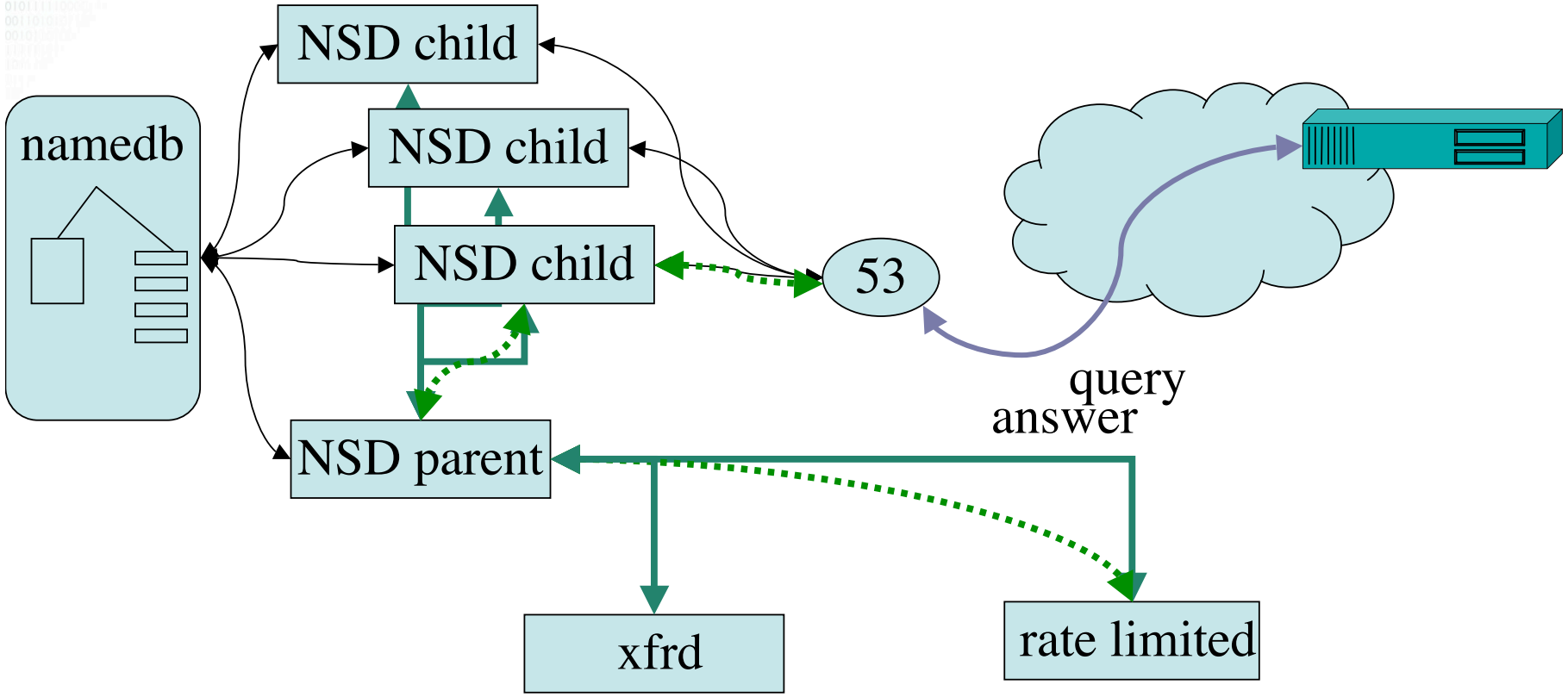
Merge diffs into zonefile using a patch utility
Recompile using zonec

All happens outside the nsd processes

Allows for off-line "cleanup"

NSD3 Architecture

Possible DOS handling



Rate limited process takes all 'difficult traffic' (e.g. NSEC3 calculation)

Rate limiting has not been implemented

- Rate limiting by moving all data over IPC might be more expensive than handling the packet by the clients directly
  - Performance measurements will help us decide
  - Not implemented in 3.0
- Other details of the NSD 3.0 may also be subject to change
- First public beta of NSD3.0 somewhere in Q3

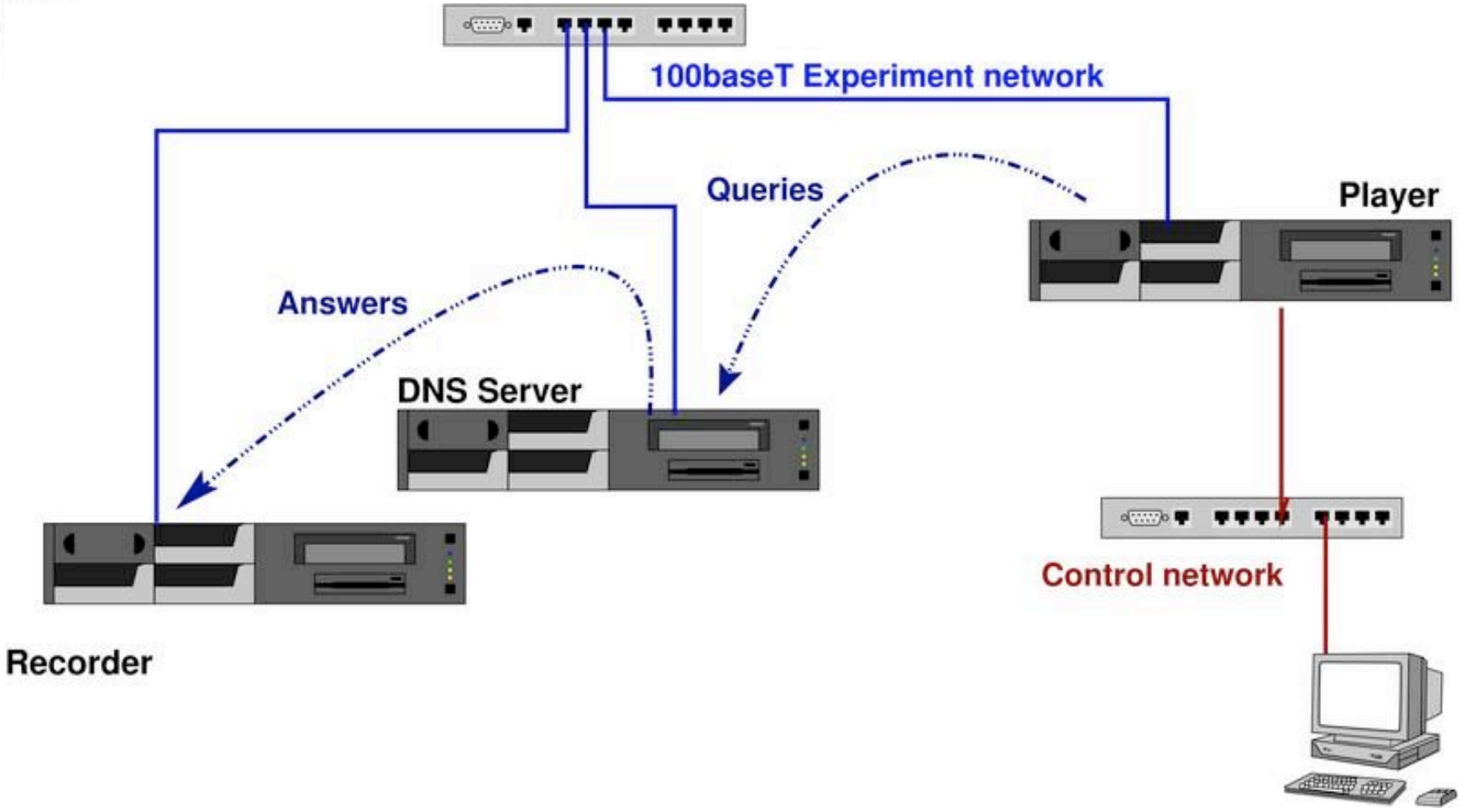
Outline

- Background on NSD: what, when, who
- Design and Architecture: goals and discription
- NSD3
- DISTEL: Regression and Performance

Distel Testlab

- Developed by Daniek Karrenberg (RIPE NCC) as part of the NSD project
- Using production zones and real-time query load
- Performance
  - Replaying traces in real time, accelerated and delayed
- Regression
  - Understanding differences with various implementations

The "DISTEL" Test Lab



DISTEL properties

- Player plays libpcap traces in real time
  - libpcap traces are modified to have the servers destination address
  - Needed modified **`tcpreplay`** to get to ms timing precision
- Server has a default route to the recorder
- Recorder captures answers
- 2 Ghz Athlon based hardware with 1 Gb memory and 100baseT Ethernet

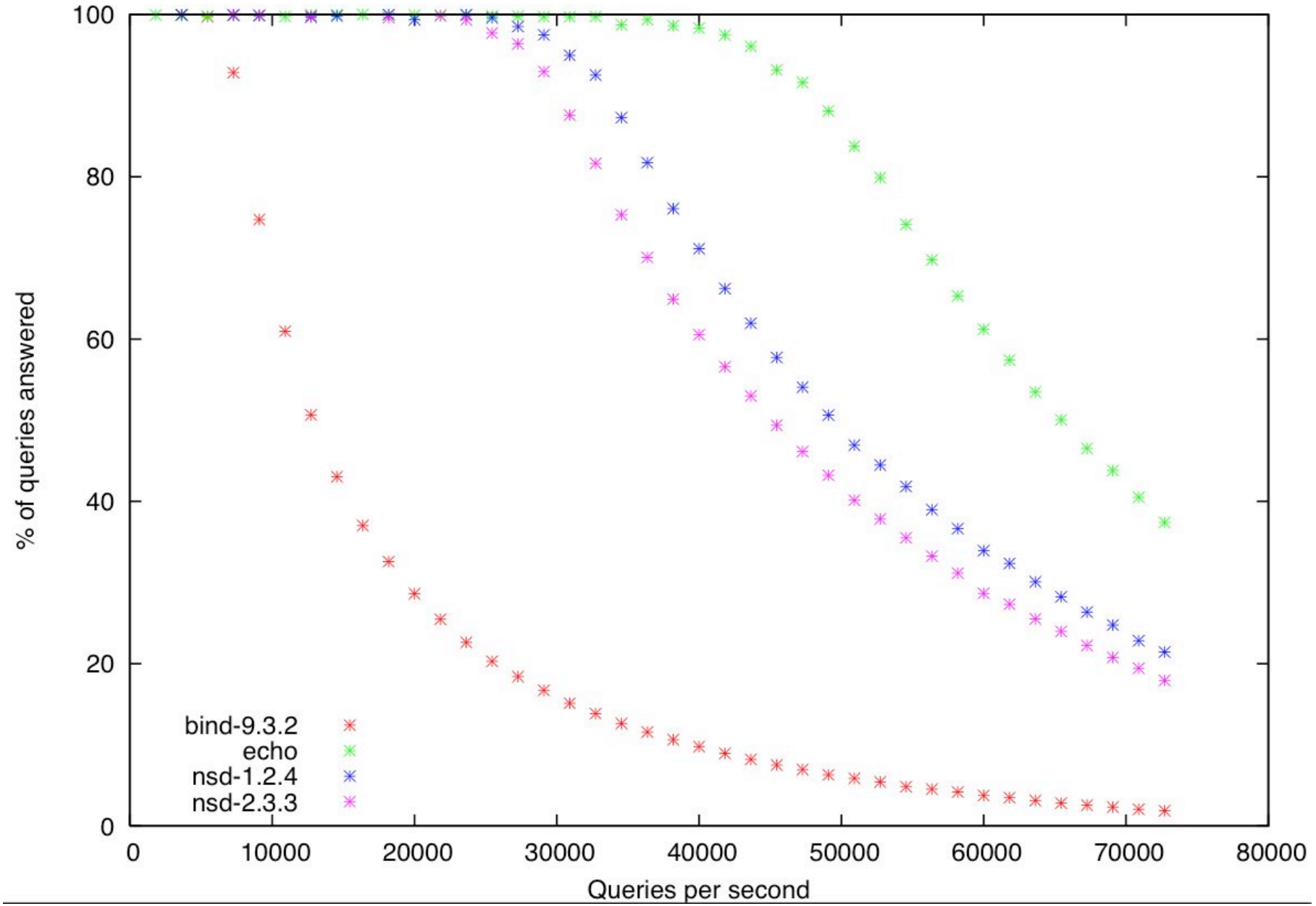
DISTEL shortcoming

- DISTEL only reports features that are present in a zone and are triggered by provided queries
  - We perform separate tests, but we may not be complete with respect to corner cases
  - It happened before and it will happen again
- You can help provide zone content and query traces
  - High volume traces, zone content you had problems with in other implementations
  - Useful for regression testing

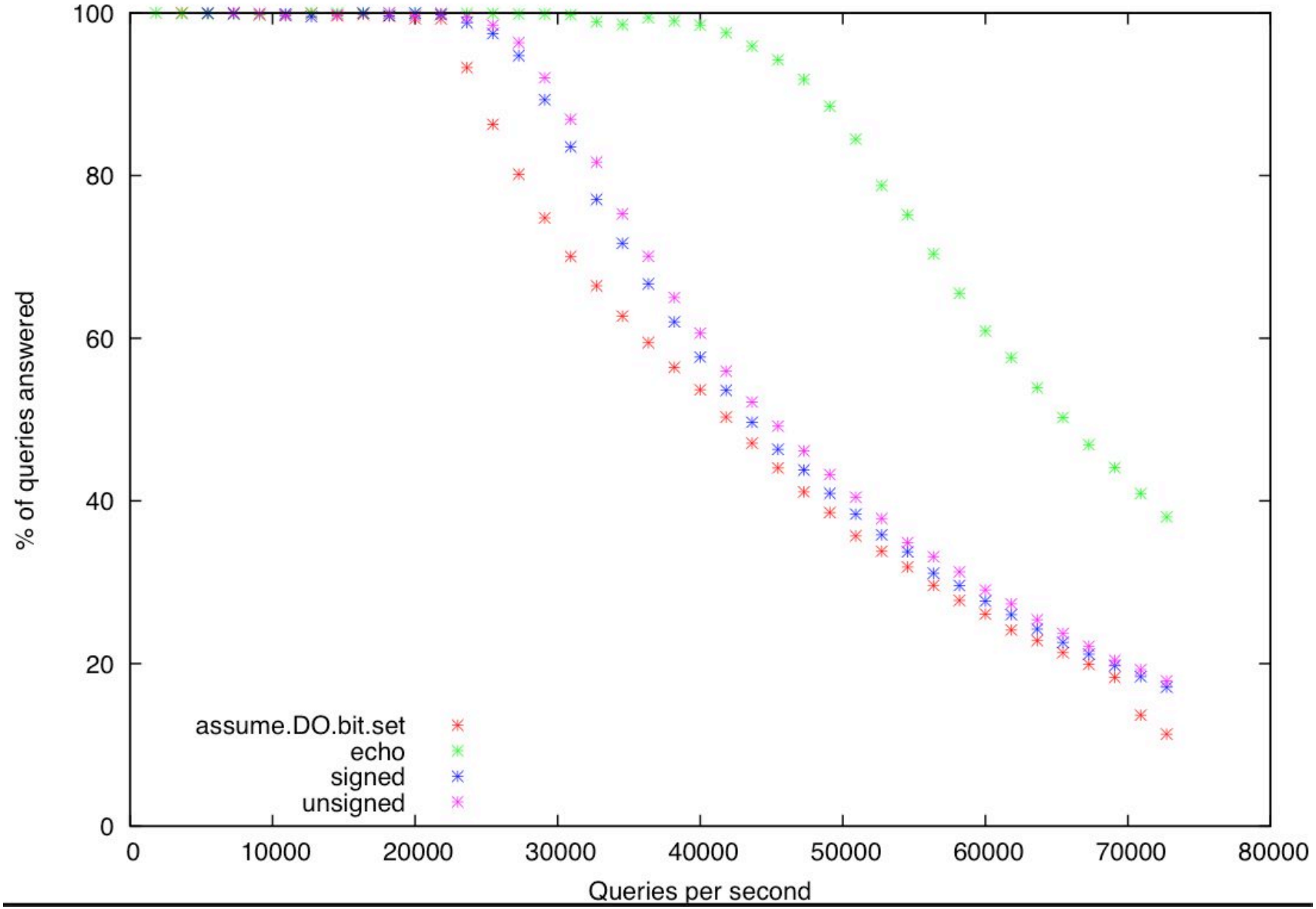
"Distel Demo"

- Using a query trace captured from k.root-servers.net agains the test server configured as k.root-server.net
  - NB: not the same hardware specs as the "real" thing
- Comparing unsigned, signed and worse case
  - Number of DO bits set in the query streams
- Read RIPE 352 for more details

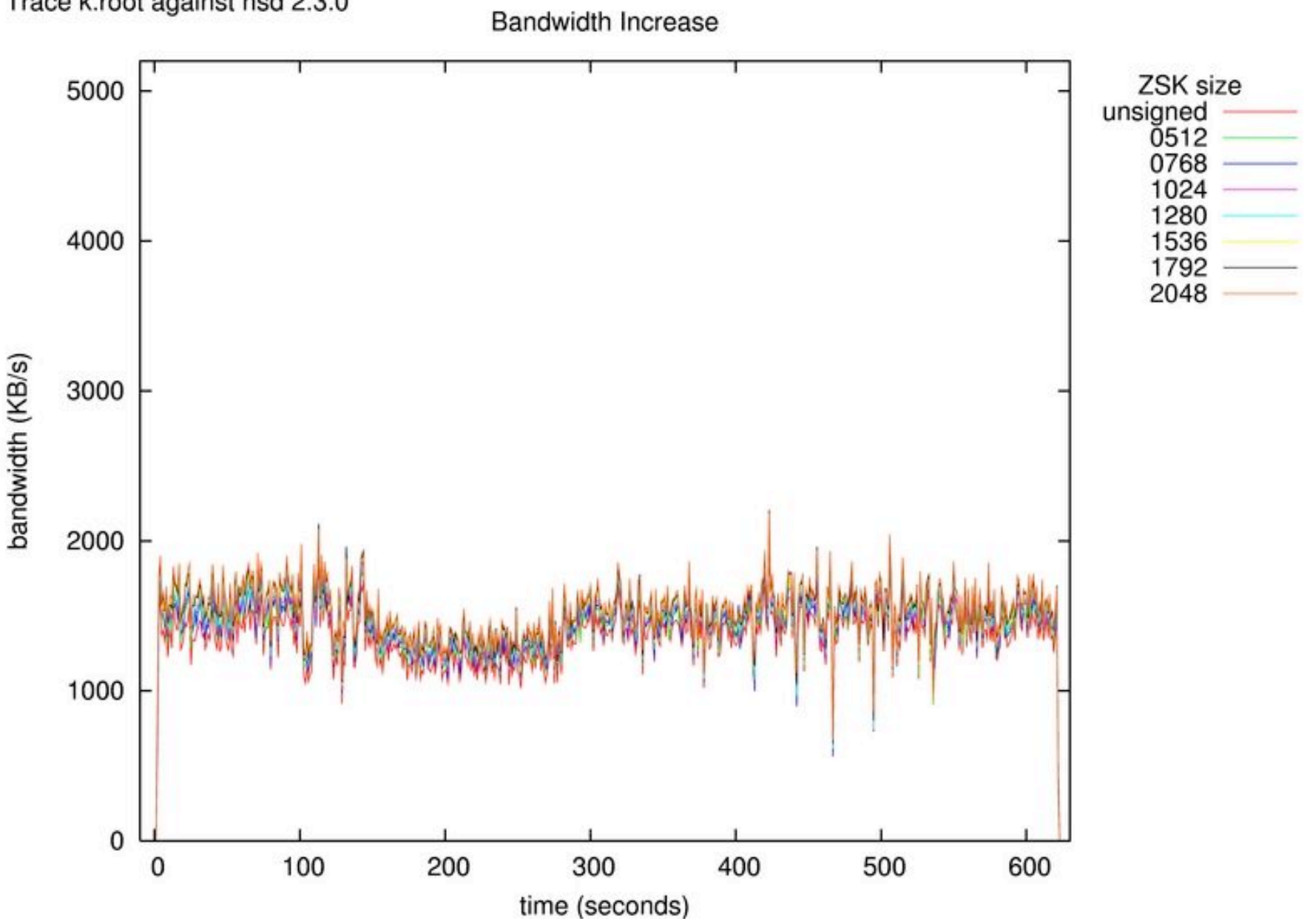
NSD 2.3.3 performance (root-zone, k-root traces)

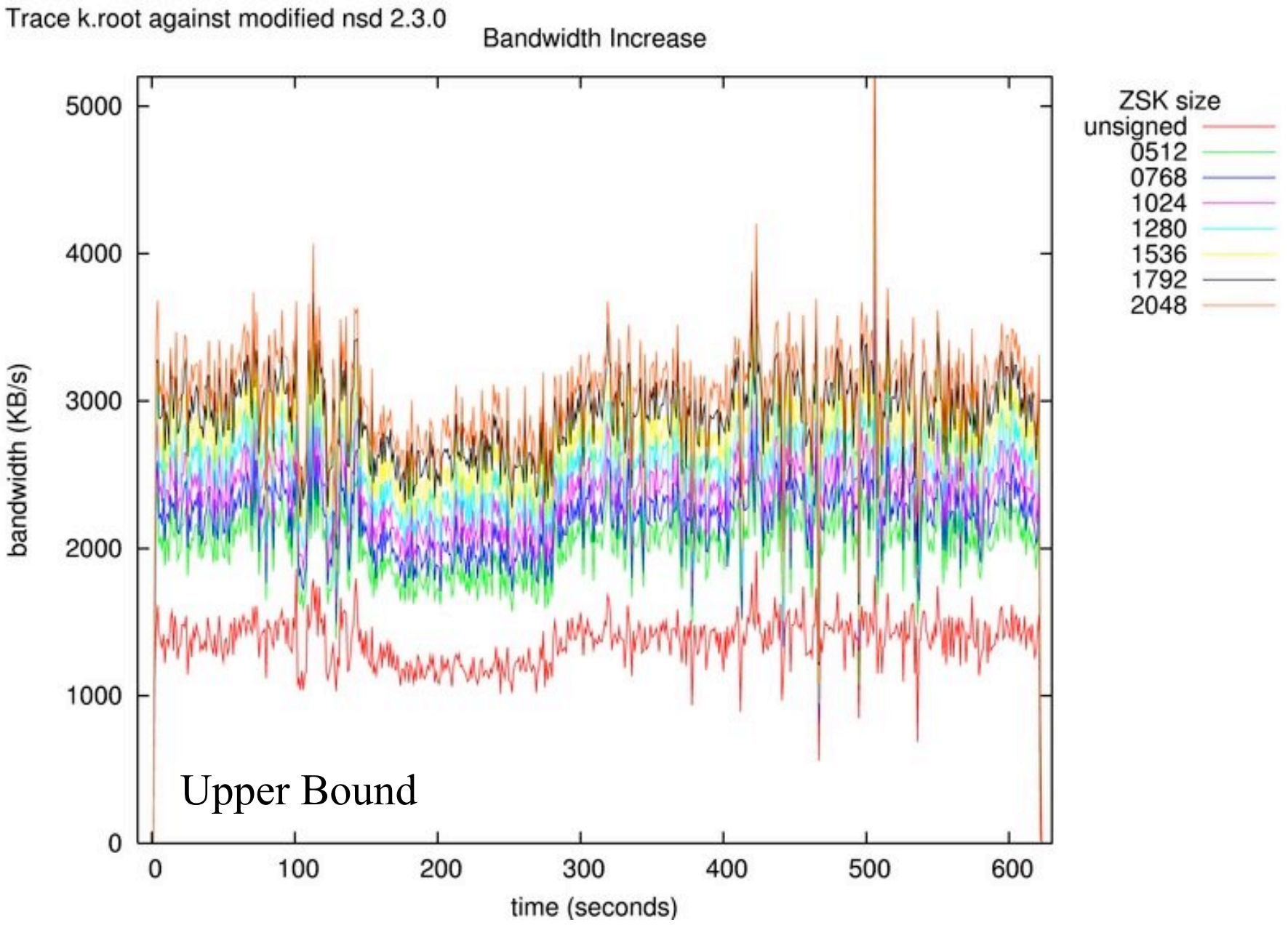


NSD 2.3.3 effect of signed zones (root-zone, k-root traces)



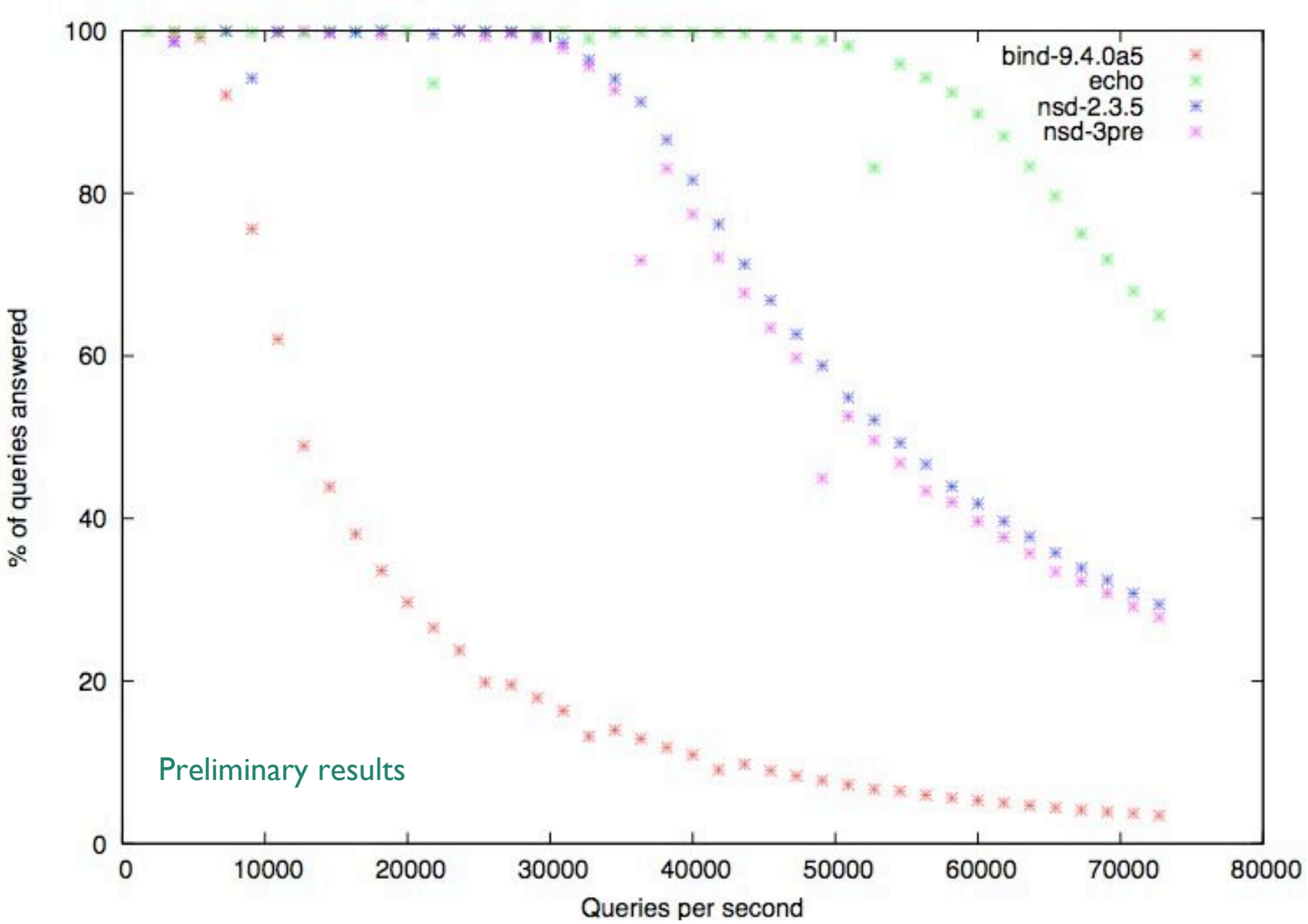
Trace k.root against nsd 2.3.0




Trace k.root against modified nsd 2.3.0
Bandwidth Increase
bandwidth (KB/s)
time (seconds)
0
1000
2000
3000
4000
5000
0
100
200
300
400
500
600
ZSK size
unsigned
0512
0768
1024
1280
1536
1792
2048
Upper Bound

New !!! More Bits

- The labs is being ported to a 1Gb network
- 1st test ran last night
  - These are preliminary results
  - There are some 'dips' that are not implementation specific and have to do with effects in the lab itself
- We have not yet explained the differences in speed
  - Could be more efficient interfaces
  - Could be nsd filled the network


bind-9.4.0a5
echo
nsd-2.3.5
nsd-3pre
% of queries answered
Queries per second
0
20
40
60
80
100
0
10000
20000
30000
40000
50000
60000
70000
80000
Preliminary results

How can you help?

- Provide zone content and query traces
  - High volume traces, zone content you had problems with in other implementations
  - Useful for regression testing
- Use the program
  - Report bugs, omissions in documentation, etc

Support

- NLnet Labs supports NSD
  - "Community support"
    - nsd-users list
    - And bugtraq
  - Two year advance notice before support is stopped
- We will also offer support contracts in the near future

Download

http://www.nlnetlabs.nl/nsd/