NSD

name server daemon

implementing a DNS server

Olaf M. Kolkman
olaf@NLnetLabs.nl

NLnet Labs

Outline

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

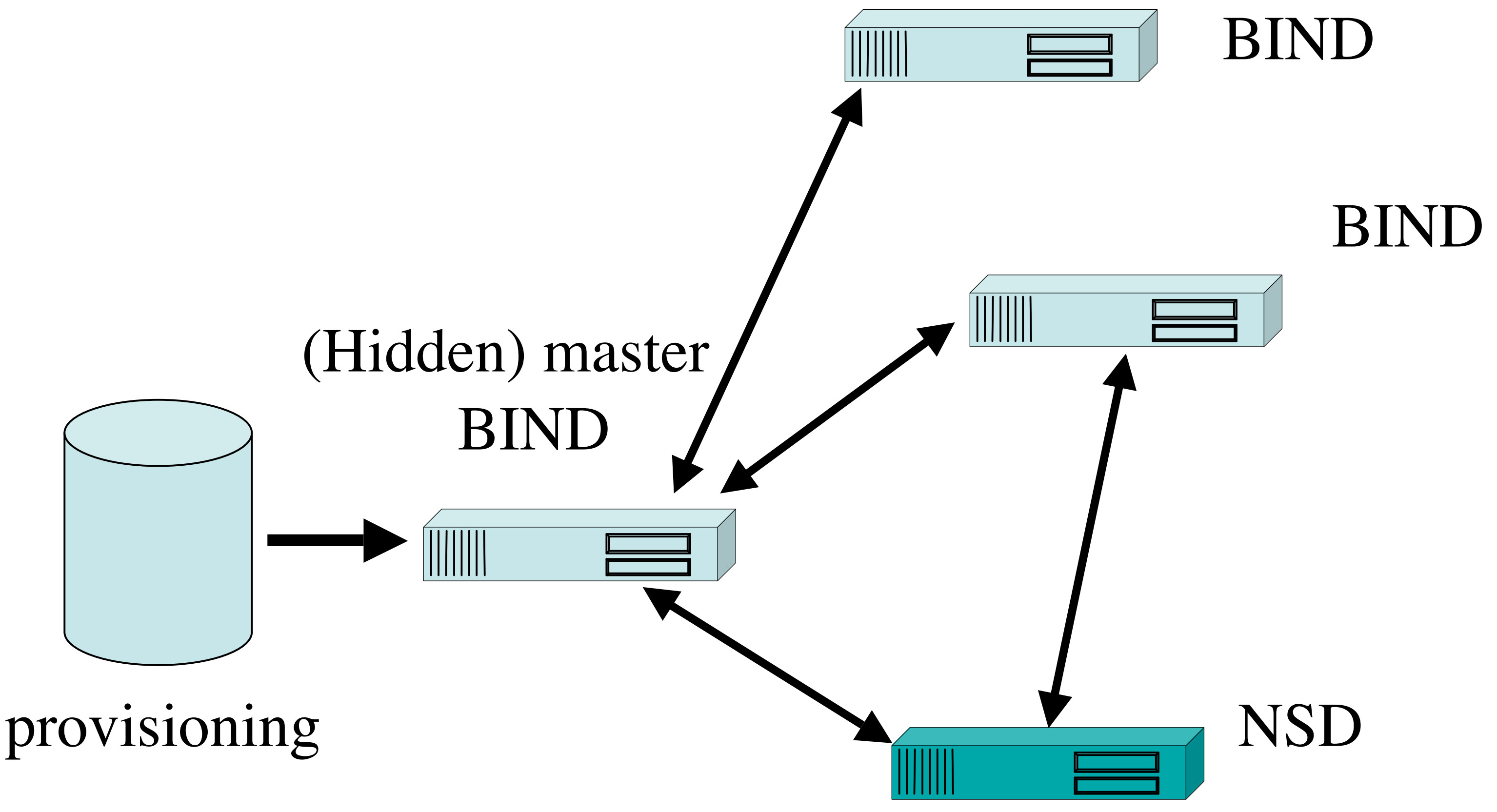
What Is NSD

- 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 and Standards 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
- 17 out of 915 TLD servers use NSD
  - According to fpdns based script
    - Ignores anycast, load balancing, changing configs
  - Include TLD servers for .NL, SE, AT, DK, CZ
  - Other TLDs have shown interest

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
- 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)

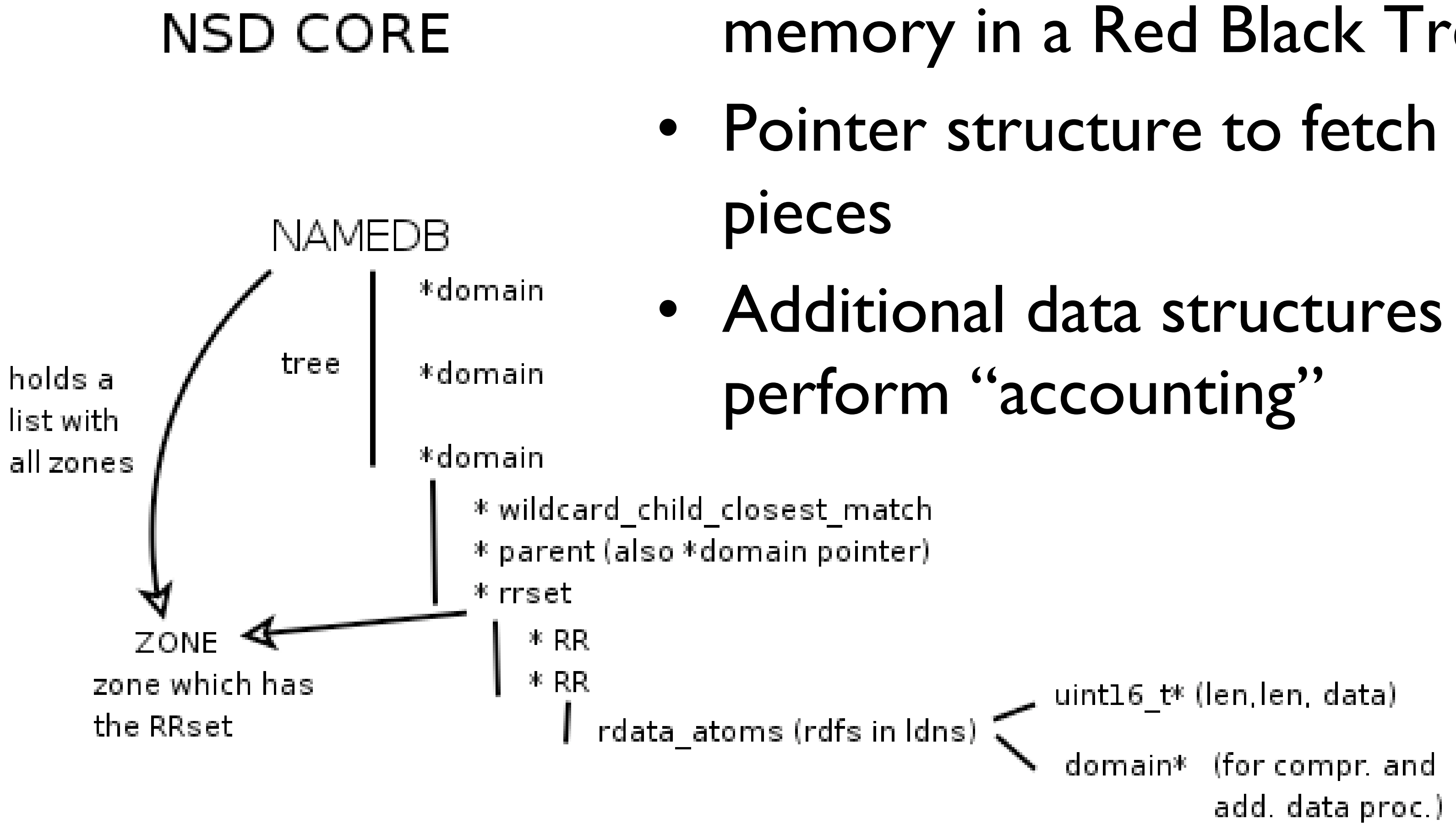
Explicit non-goals

- 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

NSD Architecture's Main Feature

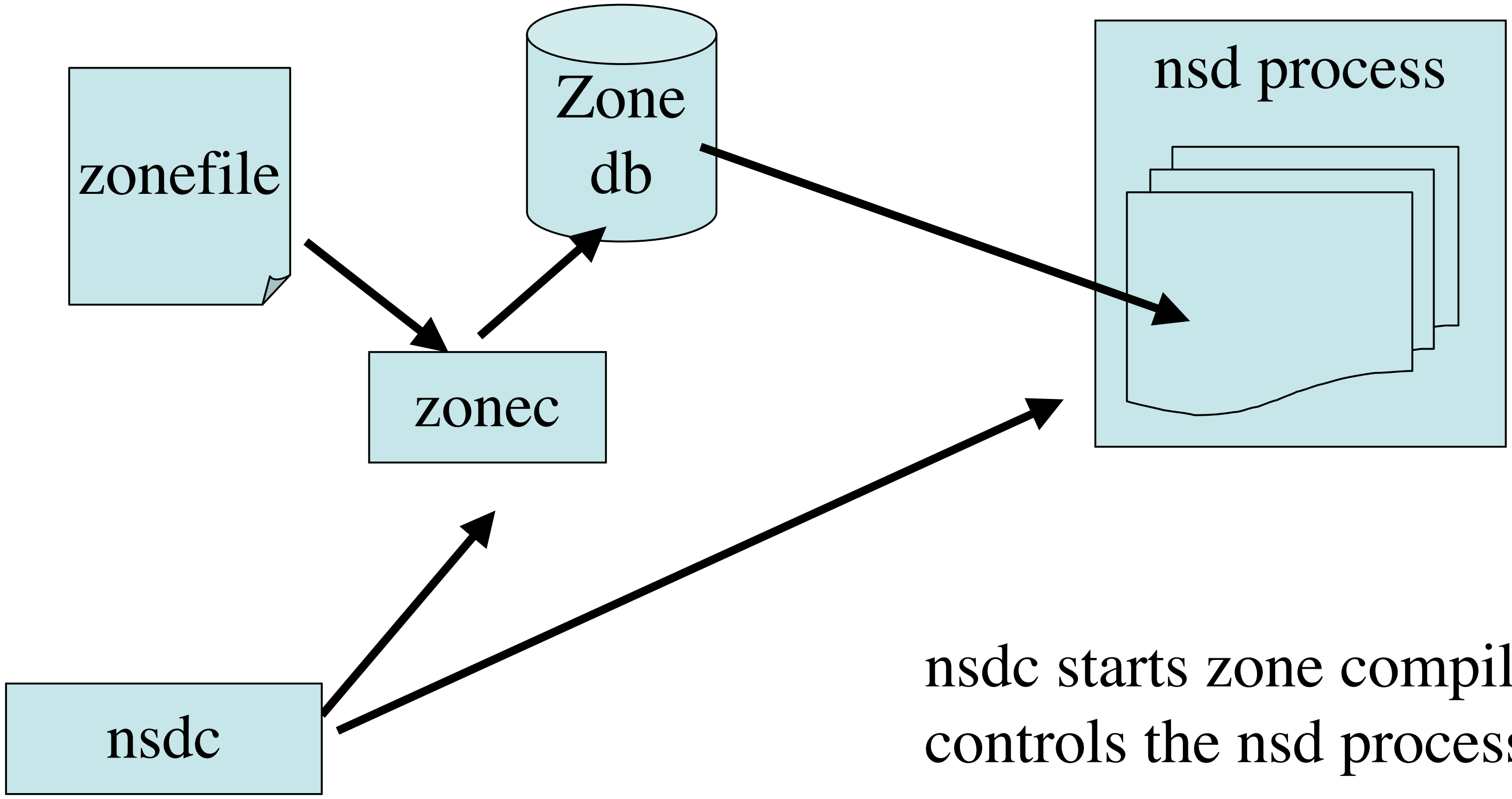
- Pre-compile answers as much as possible and perform as little work as possible during serving
  - NSD1 had fully compiled answers
    - Only some name compression at run-time
  - NSD 2 only compiled RR sets
    - Assembly at run-time to enable support of DNSSEC
    - Small performance penalty
  - NSD 3
    - In memory maintenance to support IXFR
    - Improved IPC for possible DOS handling and NSEC3 support

NSD Data


NSD CORE
NAMEDB
tree
*domain
*domain
*domain
holds a list with all zones
* wildcard_child_closest_match
* parent (also *domain pointer)
* rrset
ZONE
zone which has the RRset
* RR
* RR
rdata_atoms (rdfs in ldns)
uint16_t* (len,len, data)
domain* (for compr. and add. data proc.)

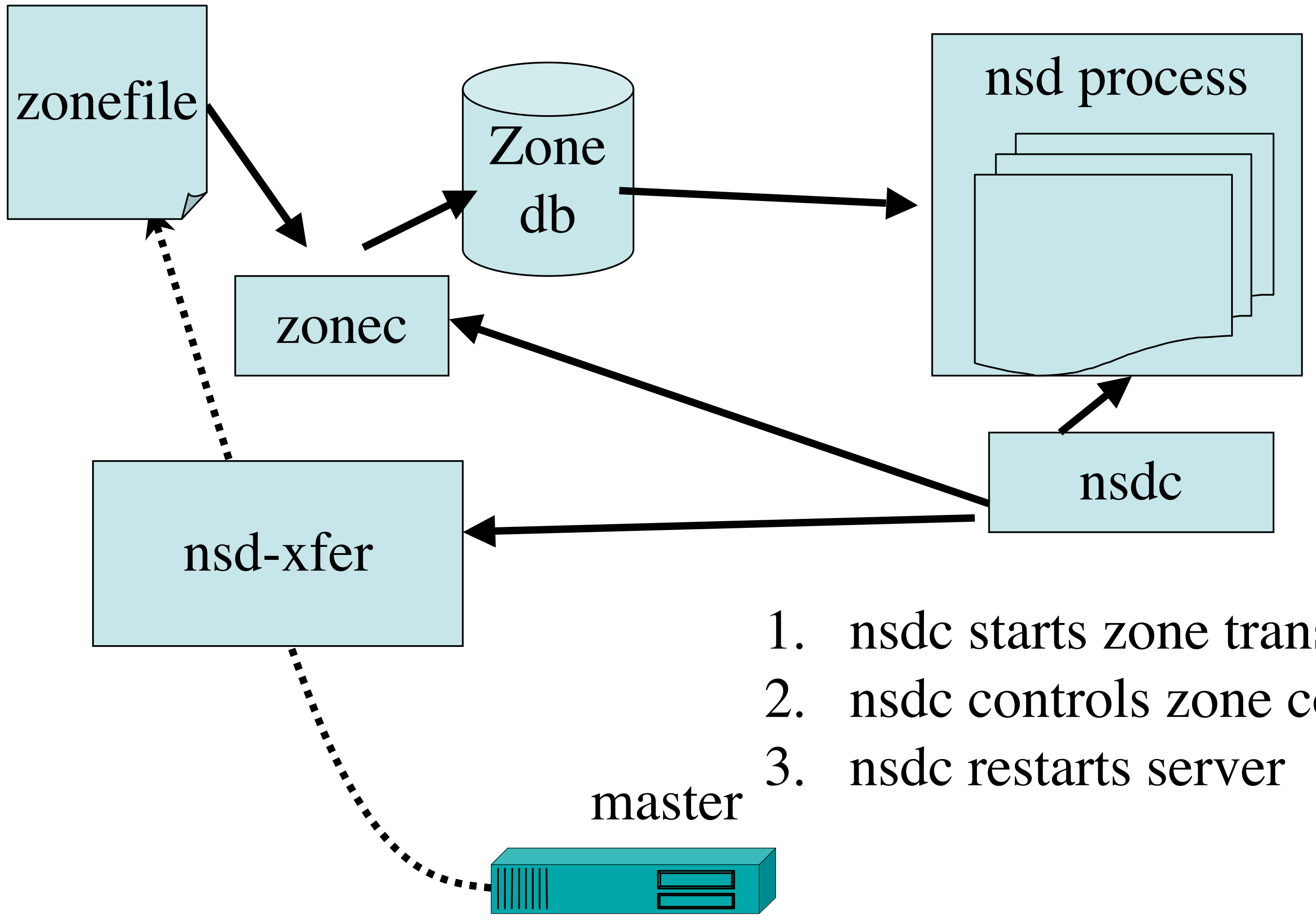
- Precompiled data stored in memory in a Red Black Tree
- Pointer structure to fetch all pieces
- Additional data structures to perform "accounting"

NSD1/2 operation model (zone loading)



nsdc starts zone compilation and controls the nsd process

NSD1/2 operation model Zone transfer

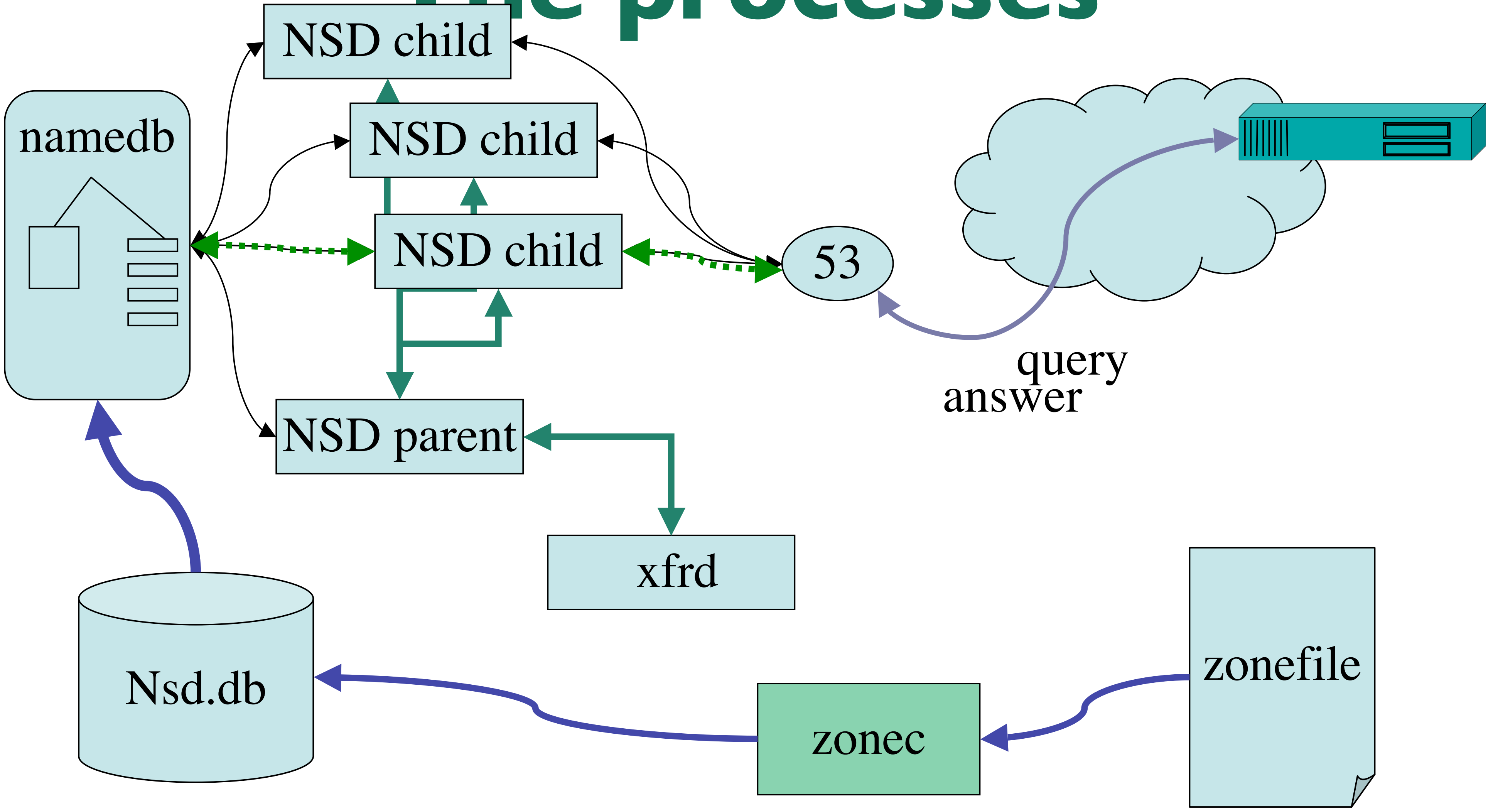


1. nsdc starts zone transfer process
2. nsdc controls zone compilation
3. nsdc restarts server

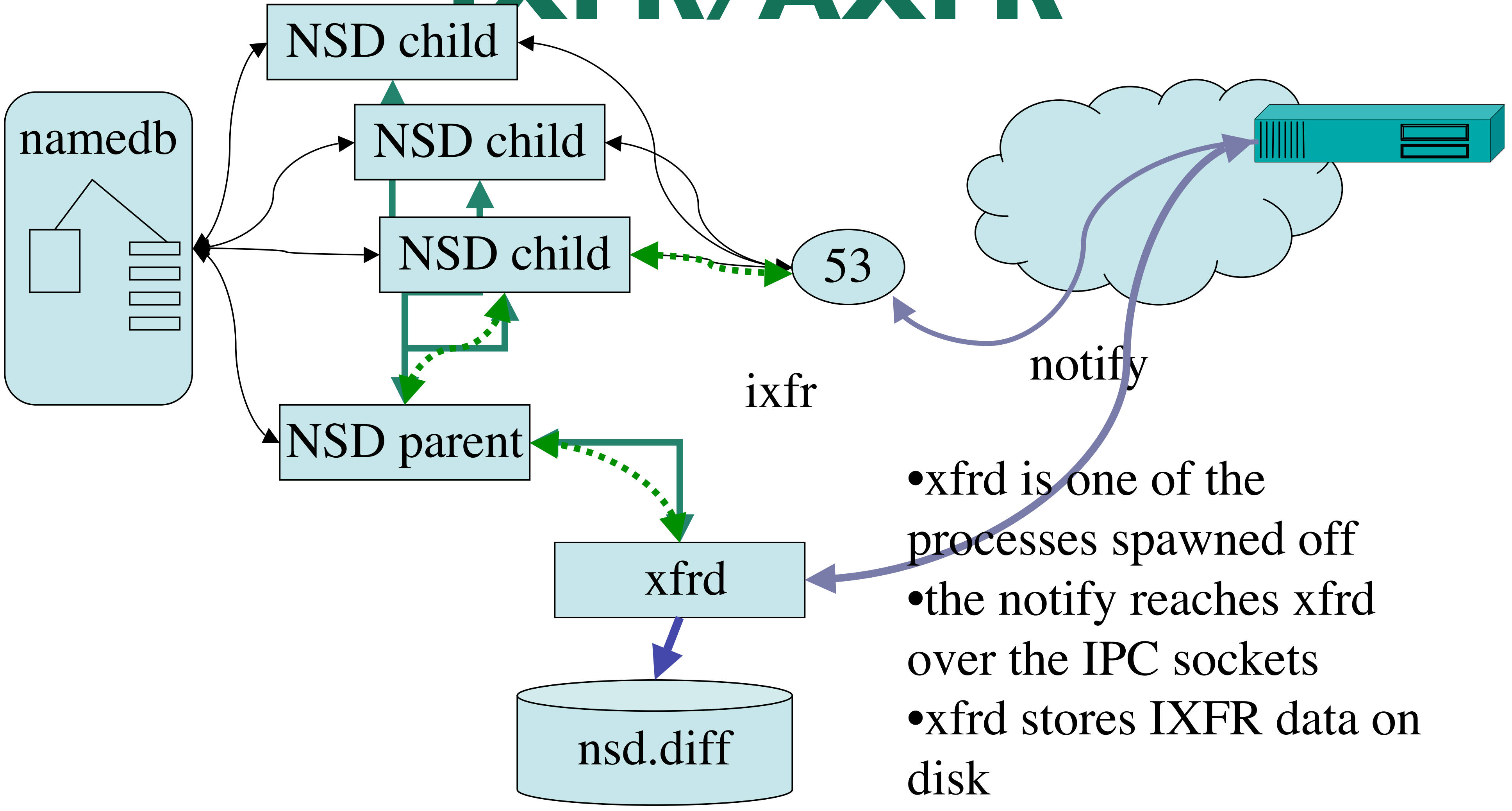
NSD 3
New Features

- 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

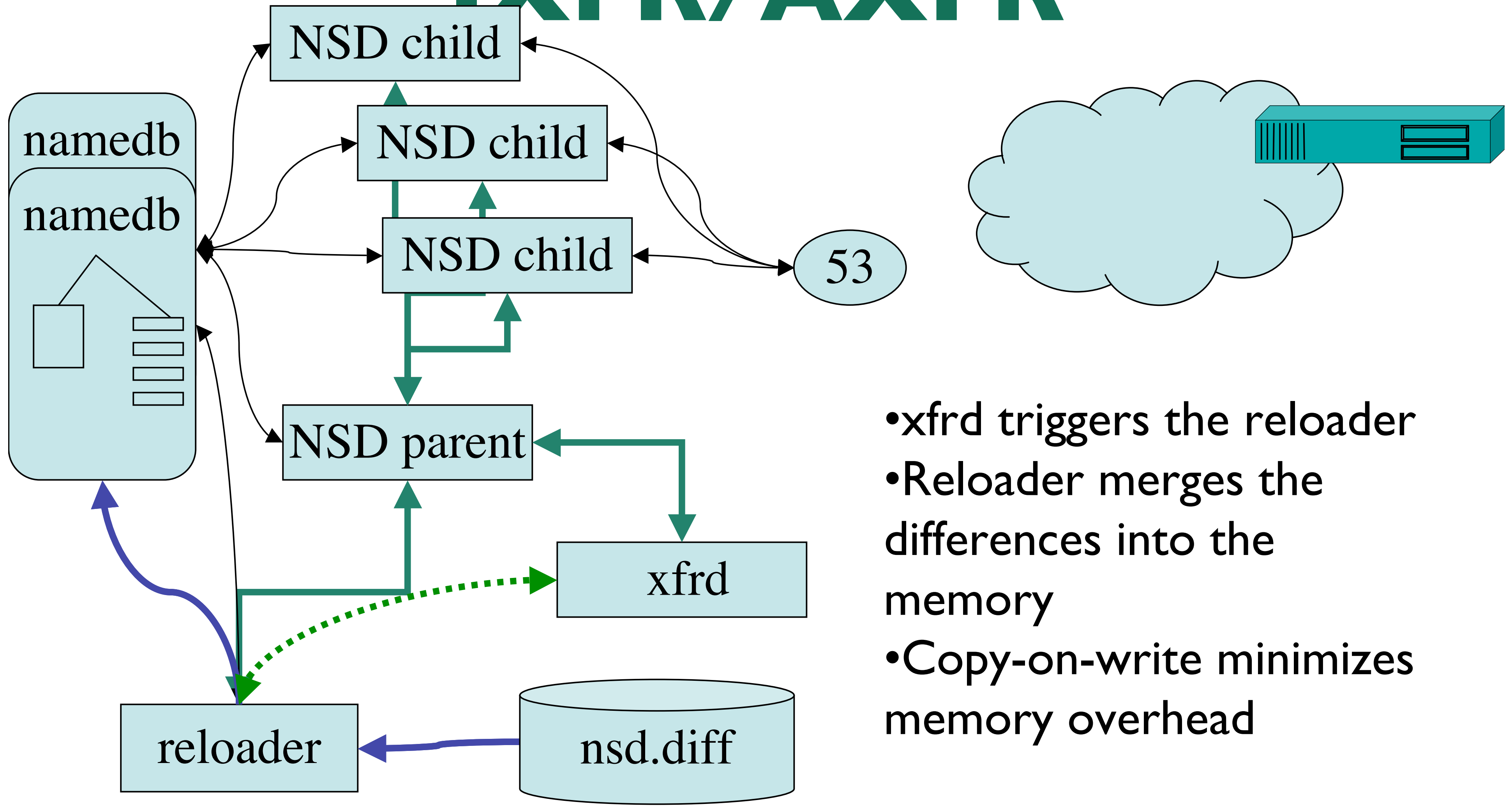


NSD3 Architecture IXFR/AXFR



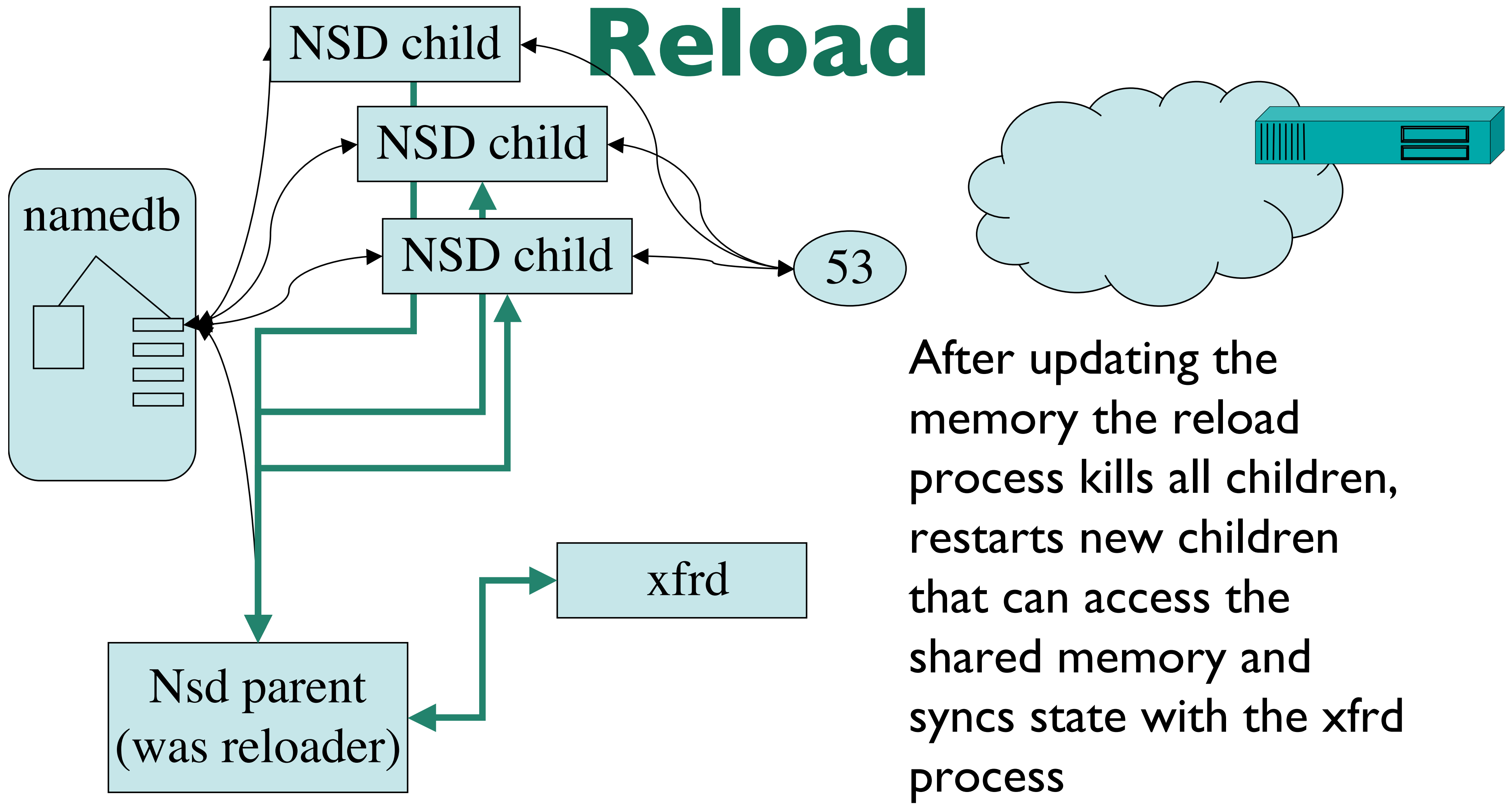
- xfrd is one of the processes spawned off
- the notify reaches xfrd over the IPC sockets
- xfrd stores IXFR data on disk

NSD3 Architecture IXFR/AXFR



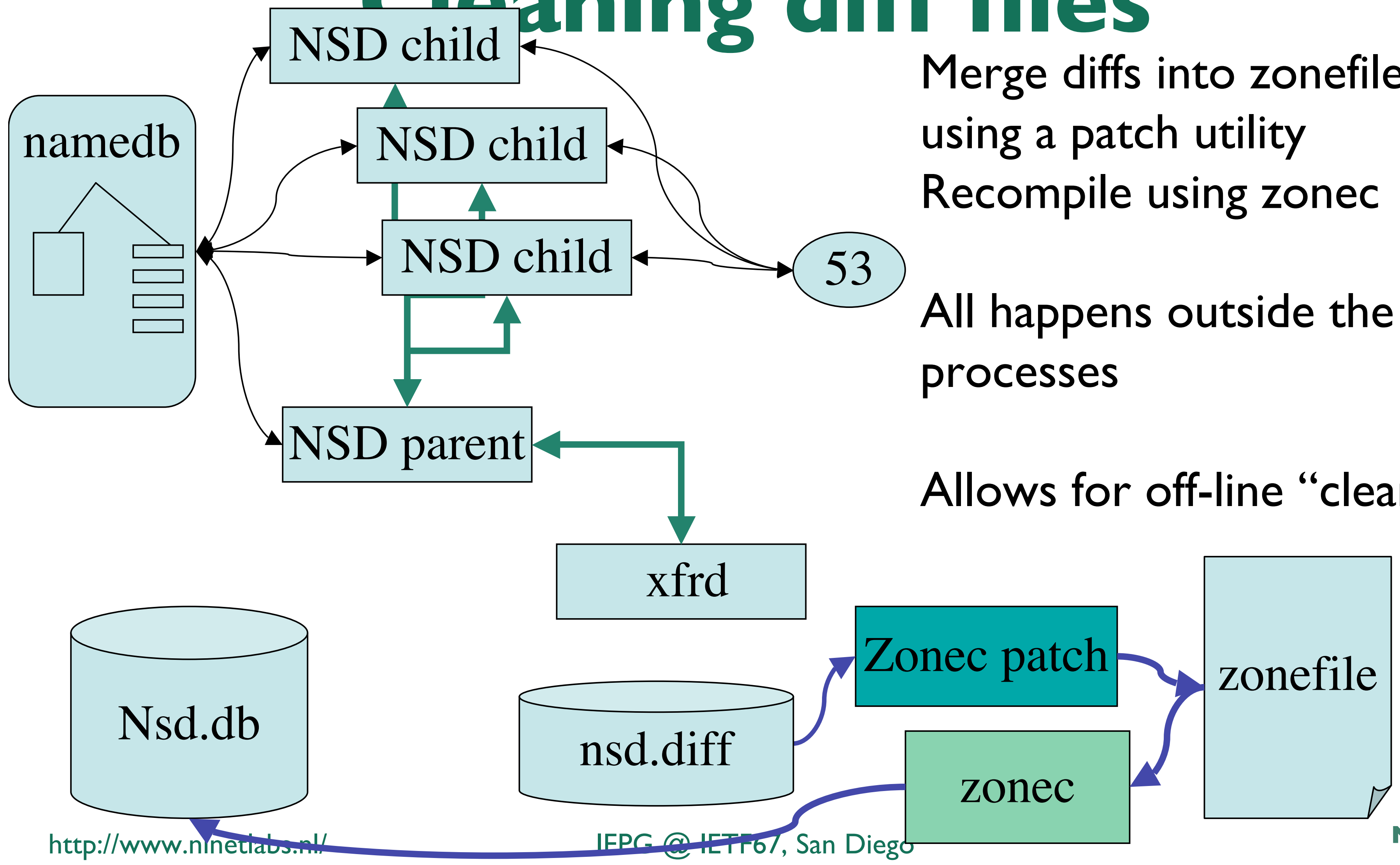
- xfrd triggers the reloader
- Reloader merges the differences into the memory
- Copy-on-write minimizes memory overhead

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



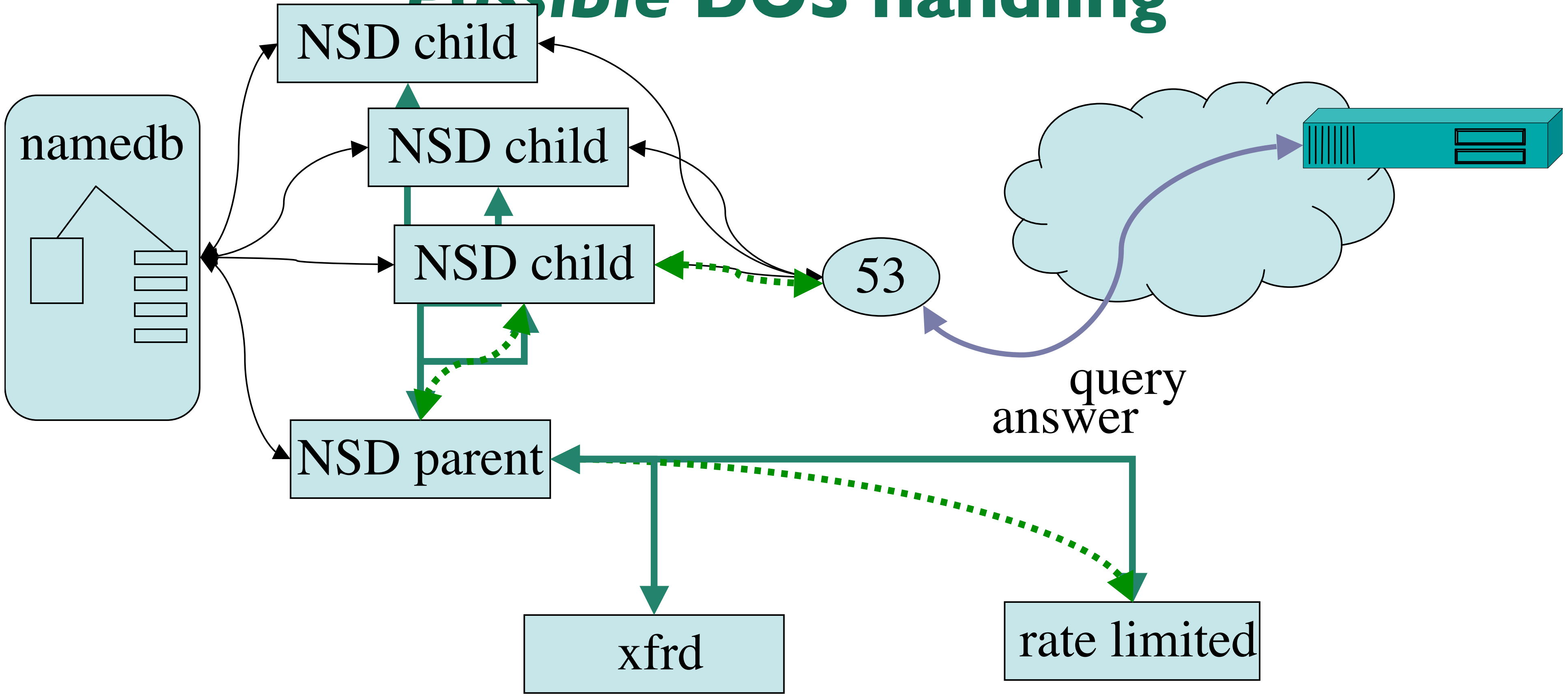
Merge diffs into zonefile using a patch utility
Recompile using zonec

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)

Not Implemented

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 3.0.0

NSD 3 releases

- NSD 3.0.0 released September 7, 2006
- NSD 3.0.1 released September 9, 2006
  - Fix of a minor but critical problem with the patch code.
- NSD 3.0.2 released November 3, 2006
  - Improves memory management; relevant for larger zones
    - .SE registry has been extremely helpful in analyzing this
  - Better portability
  - Minor bugs

Outline

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

NSD testing

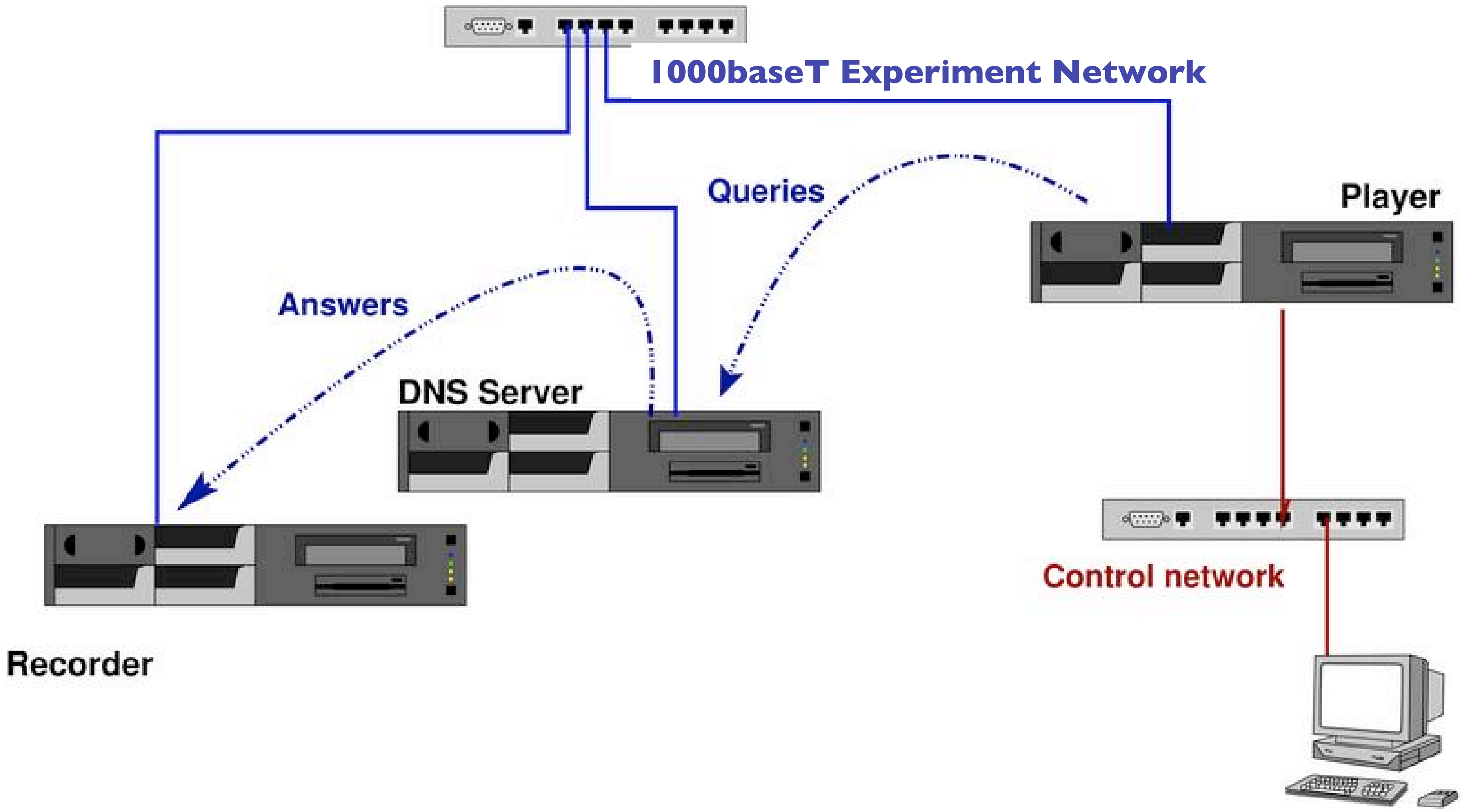
- After each SVN check-in
  - Unit Tests
    - Checks 130 assertions
  - Functionality and prev bugs tests
    - 66 scripts
  - Regression Tests
- Manual testing
  - Take to long or need special permissions
    - 19 tests

- DISTEL based test
  - Regression
  - Performance

Distel Testlab

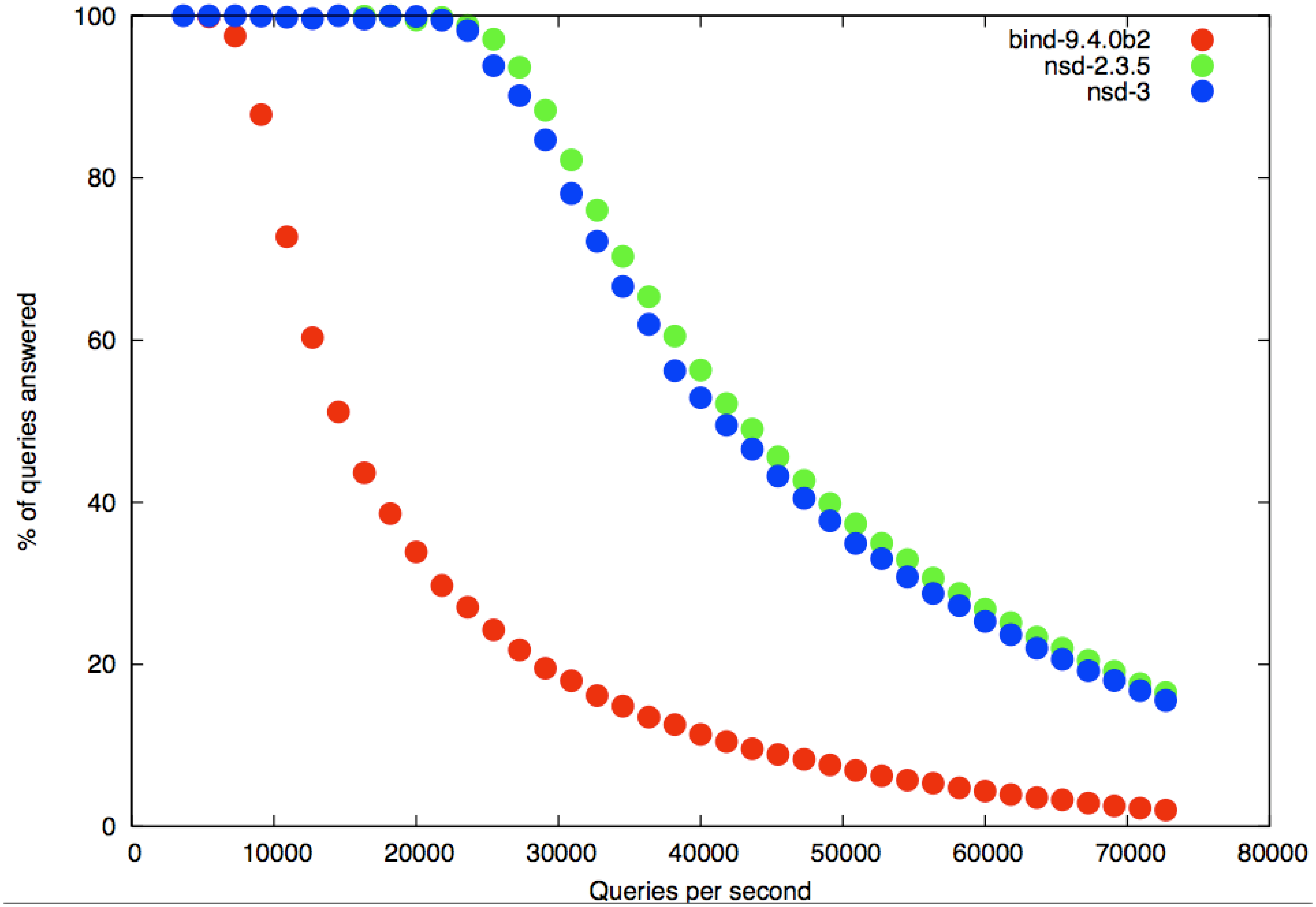
- Developed by Daniek Karrenberg (RIPE NCC) as part of the NSD project
  - Version "2" based on ldns build during NSD3 development
- Using production zones and real-time query load
- Performance
  - Replaying traces in real time, accelerated and delayed
- Regression
  - Understanding differences with 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 1000baseT Ethernet


bind-9.4.0b2
nsd-2.3.5
nsd-3
% of queries answered
Queries per second
0
20
40
60
80
100
0
10000
20000
30000
40000
50000
60000
70000
80000

Comparison between NSD 3.0.0 and BIND 9.3.2 for a root trace.

```
Difference                        packets      fraction
d-additional (2.4.5)                    455607       59.19%
d-clrdobit (2.3.1)                208389       27.07%
b-soattl (2.3.5)                  101707       13.21%
n-update (2.4.2)                  1858         00.24%
d-hostname (2.4.7)                1032         00.13%
d-formerrquery (2.4.9)            773          00.10%
b-class0 (2.3.3)                  264          00.03%
d-refusedquery (2.4.6)            79           00.01%
d-notify (2.4.1)                  18           00.00%
b-mailb (2.4.3)                   7            00.00%
n-tcinquery (2.3.4                6            00.00%
b-classany-nxdomain (2.3.6)       5            00.00%
d-badquery?ags (2.4.10)           4            00.00%
n-ixfr-notimpl (2.4.8)            3            00.00%
d-version (2.4.4)                 1            00.00%
Total                             769753        100%
Number of packets the same after normalization: 1474863
Number of packets exactly the same on the wire:   59161
Total number of packets inspected:              2244616
```

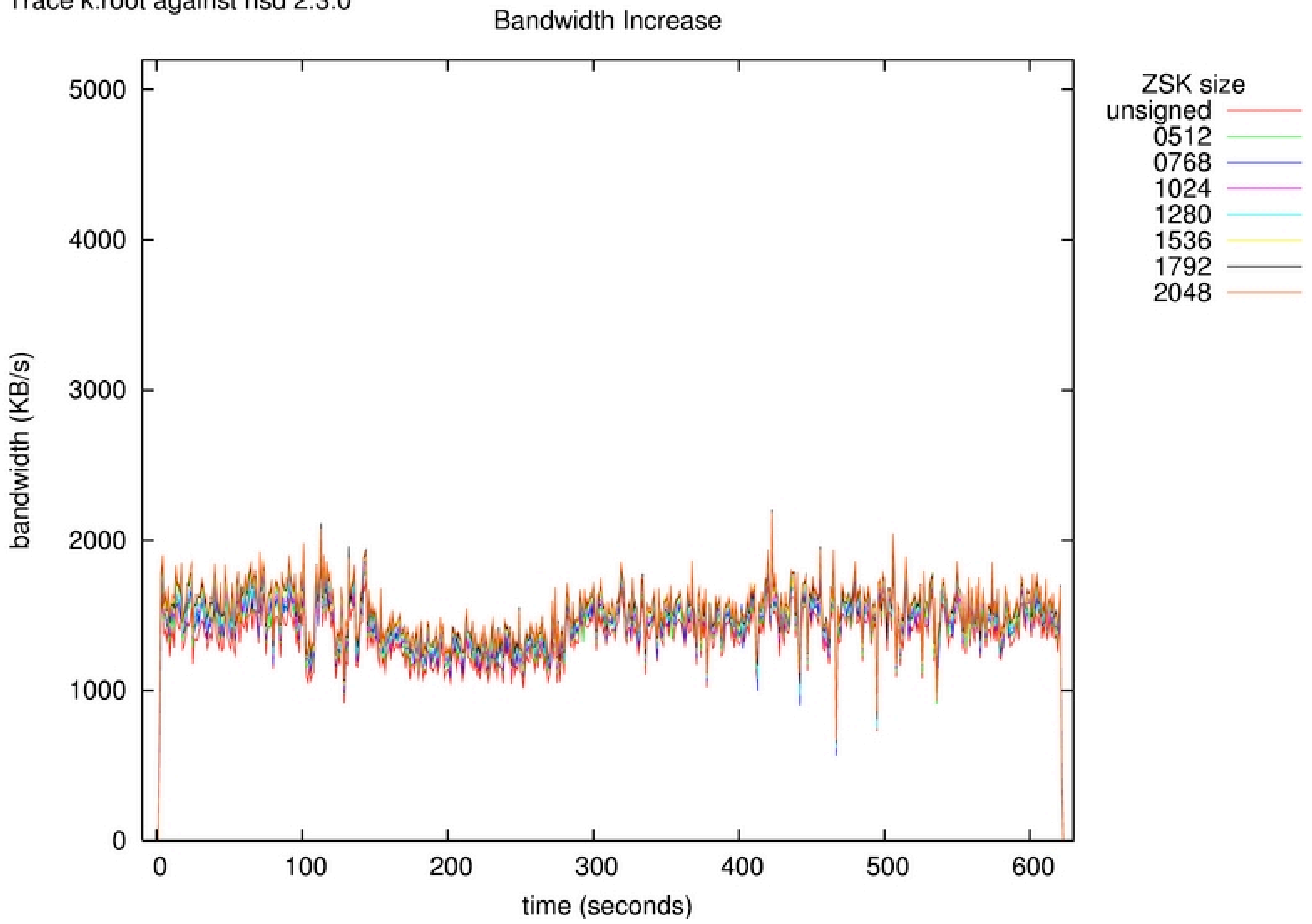
DISTEL shortcoming

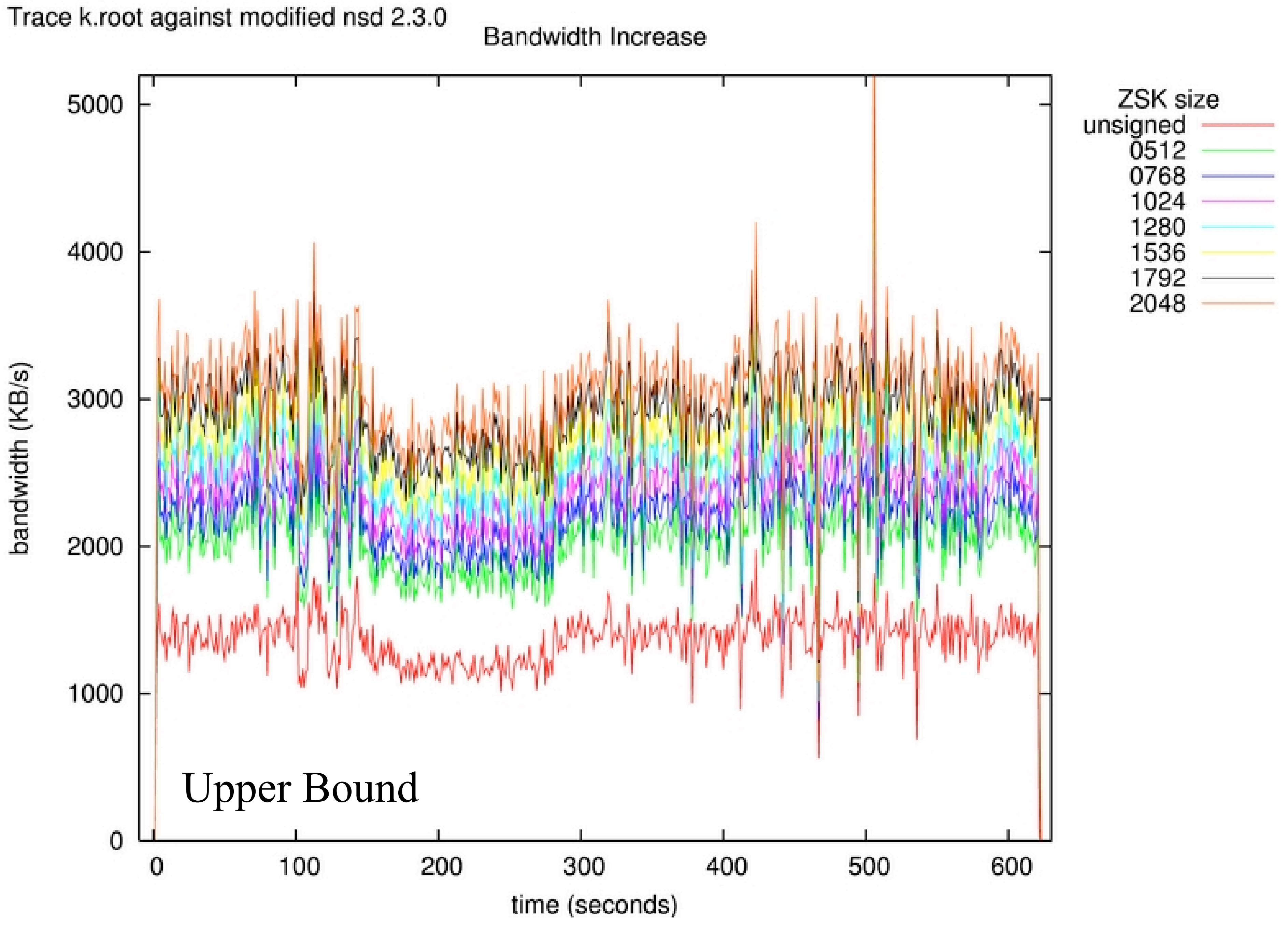
- DISTEL only reports features that are present in a zone and are triggered by provided queries
  - We perform separate tests, but we may not be complete with respect to corner cases
  - It happened before and it will happen again
- You can help provide zone content and query traces
  - High volume traces,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

Distel as R/D tool

- 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

Trace k.root against nsd 2.3.0




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

What did we learn/ How can you help?

- As developer it is extremely difficult to realize what the true operational problems are
  - One of the causes of underestimating the memory problems that have been solved in 3.0.2
- 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
  - Help us understand your operational environment

Support

- NLnet Labs supports NSD
  - “Community support”
    - nsd-users list
    - And bugtraq
  - Two year advance notice before support is stopped
    - NLnet Labs expects to be around until at least 2015
- NSD Support contracts
  - See www.nlnetlabs.net/nsd/support.html
- Download

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

Questions?

LEFT OVERS

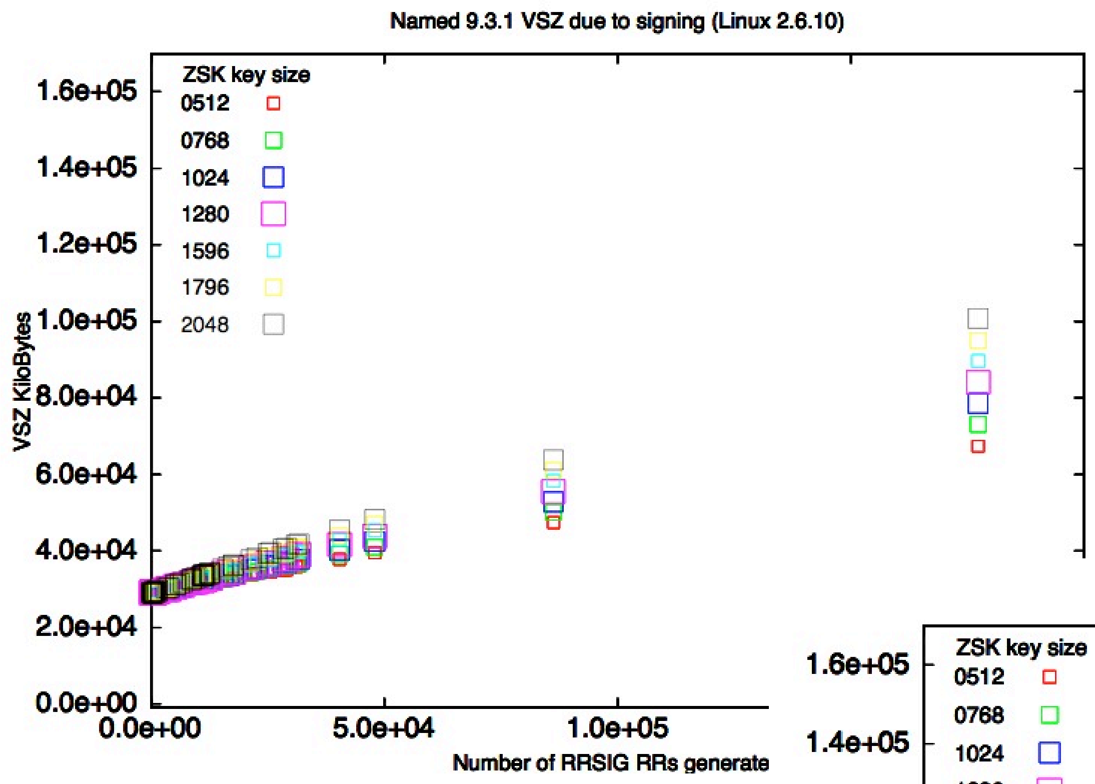
NSD 2 Operational Features

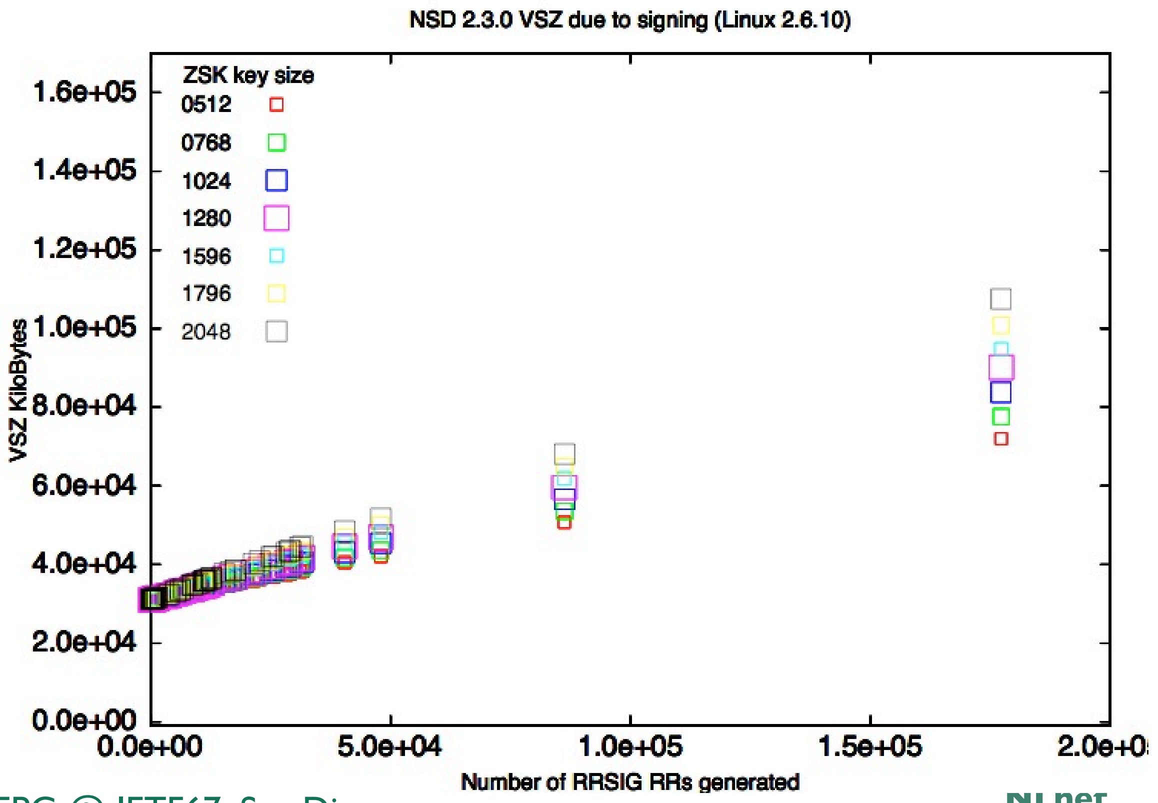
- Requires 'cron' and/or manual control for ingress zone transfers
- .NL zone signed with 1024big ZSK

	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


Named 9.3.1 VSZ due to signing (Linux 2.6.10)
ZSK key size
VSZ KiloBytes
1.6e+05
1.4e+05
1.2e+05
1.0e+05
8.0e+04
6.0e+04
4.0e+04
2.0e+04
0.0e+00
0.0e+00
5.0e+04
1.0e+05
Number of RRSIG RRs generate


NSD 2.3.0 VSZ due to signing (Linux 2.6.10)
ZSK key size
VSZ KiloBytes
1.6e+05
1.4e+05
1.2e+05
1.0e+05
8.0e+04
6.0e+04
4.0e+04
2.0e+04
0.0e+00
0.0e+00
5.0e+04
1.0e+05
1.5e+05
2.0e+0
Number of RRSIG RRs generated