# Distributed DNS Troubleshooting *

Vasileios Pappas
*UCLA Computer Science*
*vpappas@cs.ucla.edu*

Patrik Fältström
*Cisco Systems*
*paf@cisco.com*

Daniel Massey
*Colorado State University*
*massey@colostate.edu*

Lixia Zhang
*UCLA Computer Science*
*lixia@cs.ucla.edu*

## ABSTRACT

In this paper we present a troubleshooting tool designed to identify a number of DNS configuration errors. These errors range from commonly seen misconfigurations that are well known among DNS operators, such as lame delegations, to less known ones, such as cyclic zone dependencies. Left unnoticed, these misconfigurations can seriously affect the availability of the DNS infrastructure. Instead of explicitly enumerating all possible configuration errors, we first identify two essential properties that characterize a correct DNS configuration, and detect misconfigurations as violations of these properties. We also utilize multiple monitoring points to identify configuration errors that are difficult or impossible to pin down with a single vantage point. Furthermore, equipped with a comprehensive graphical user interface, our tool provides network operators with a tangible view of their DNS zones' configuration and the errors that may affect their availability.

## Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*

## General Terms

Design, Reliability

## Keywords

DNS, Misconfigurations, Troubleshooting

## 1. INTRODUCTION

The Domain Name System (DNS) is one of the largest distributed systems deployed on the Internet, upon which a great number of applications rely. At the same time, it is well known, at least among the DNS community, that a number of common DNS configuration errors exist [14]. Various DNS related operational problems stem from these errors, and usually they degrade the system's availability and performance. As our recent measurements [25] show, the impact of DNS configuration errors may range from insignificant to severe, and many of these errors considerably reduce the service availability of DNS zones and degrade the system performance by orders of magnitude.

So far the main effort in addressing the problem has focused on informing the operators about the existence of DNS configuration errors, either by Internet RFCs [14, 20] or with directives set by specific organizations [21]. There have also been a few efforts that try to detect specific DNS configuration errors appearing in the Internet [18, 22, 17], and a few DNS diagnosis tools [11, 3, 5, 4, 2] that can help detect known configuration errors for a given zone. Despite all the existing efforts, DNS configuration errors are still widespread today. We believe that one of the main reasons is the lack of adequate tools to help DNS operators monitor and identify configuration errors in their *own* domains.

This paper presents a DNS troubleshooting tool, currently under development, that aims at detecting DNS configuration errors which may affect the availability of the DNS infrastructure. Our tool differs from all the existing efforts in the following ways:

- It detects configuration errors by identifying violations of a correct configuration, rather than only checking some known errors.
- It leverages multiple monitoring points to identify errors that are difficult or impossible to detect from a single vantage point.
- It provides a comprehensive user interface, which allows operators to get a tangible view of their DNS zones and the errors that may affect system availability.

The rest of the paper is structured as follows. In section 2 we identify two properties that characterize a correct DNS configuration, and we show by example how a number of DNS configuration errors can be detected as violations of those properties. In section 3 we give a high level description of our troubleshooting tool system architecture and in section 4 we describe the specific diagnostic tests currently implemented by the tool. Section 5 provides a few further details about the system implementation and usage. Section 6 compares our effort with a number of related works, followed by the conclusion in section 7.

## 2. DNS CONFIGURATION ERRORS

Developed in the 1980s, the primary goal of the Domain Name System (DNS) [23, 24] is to provide a robust and scalable name-to-

*This work is partially supported by the National Science Foundation under Contract No SCI-0353259. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of NSF.


*SIGCOMM'04 Workshops,* Aug. 30+Sept. 3, 2004, Portland, Oregon, USA.

address mapping service. DNS data is stored using a simple data structure called a *Resource Record* (RR). A RR can be viewed as a tuple $< name, TTL, class, type, data >$, where $name$ is a fully qualified domain name (FQDN) such as *www.google.com*, $TTL$ is the time-to-live value in seconds, $class$ is typically Internet (IN), $type$ identifies the resource record format, and $data$ contains the format-specific data. Examples of common RR types include A RRs that store IP addresses, MX RRs that identify mail servers, PTR RRs that map IP addresses to DNS names, and NS RRs that store name server information. DNS makes no distinctions between different types of data records, such as MX RRs used by end-user applications, and *infrastructure* RRs, such as NS RRs, used only to establish and navigate the DNS hierarchy.

In this work, we focus on identifying errors that can affect the DNS system availability, rather than examining all types of errors that may exist in the global DNS database. For example we do not check whether an MX RR is correctly configured to point to a valid SMTP server, or whether an A RR has a companion PTR RR, because such errors, if they exist, do *not* affect the availability of the DNS system, although they may affect specific applications. Instead we only pay attention to errors associated with the DNS infrastructure RRs, such as NS and glue A RRs; an improperly configured NS or glue A RR can affect DNS service availability. As DNSSEC [12] is expected to be deployed in near future, it will introduce some additional types of infrastructure RRs, such as DNSKEY and DS RRs, that can be subject to similar types of configuration errors.

While infrastructure RRs are essential for building and navigating the DNS, they are not the only *elements* that comprise the DNS system. Equally important elements are the nameservers, which store the actual DNS records and serve the DNS clients. To better analyze the DNS availability, we may divide the DNS infrastructure elements in two classes: *physical level* elements, such a nameservers, and *logical level* elements, such as infrastructure RRs. We define a DNS *entity* as a collection of DNS infrastructure elements, which plays a particular role in the DNS system. For example an *authoritative nameserver* is a DNS entity, which is defined both in the logical level, as an NS record, and in the physical level, as a nameserver. A *delegation point* again is a DNS entity, and it is defined as a collection of glue RRs, i.e. the NS RR set, stored at both the parent and the child zone, and the companion glue A RRs.

We further identify two properties of the DNS configuration that need to hold, both at the logical and at the physical level, in order for a configuration to be correct.

- *Coherency*: If a DNS entity is declared in multiple places, either at the logical or physical level, then all the declarations must be consistent. For example, every delegation point is a NS RR set, declared in two different places at the DNS logical level: both at the parent and at the child zone. The coherency property requires the two NS RR sets to be consistent, i.e. they must point to the same set of servers.

- *Independence*: If two DNS entities, declared at either the logical or physical level, are distinct, then they must be mutually independent. For example, two distinct nameservers, which are authoritative for the same or even different zones, are DNS entities declared both at the physical and at the logical level. The independence property requires them to be mutually independent, i.e. the failure of one servers does not make the other unavailable.

Violations of any of these two properties can reduce the availability of the DNS system. Coherency violations reduce the number of redundant servers unconditionally, meaning that the number of available servers is lower than the expected, as long as the violation holds. Independence violations, on the other hand, reduce the number of redundant servers conditionally, meaning that the number of available servers is lower than the expected only when a certain condition is true, such as when a specific server is unavailable.

In the rest of this section we provide examples of configuration errors that can be detected as violations of the coherency or the independence property.

## 2.1 Delegation Inconsistency

Delegation inconsistency errors happen when there is an inconsistency between the delegation RRs declared at the parent and at the child zone. The delegation records defined at the parent are the NS records of the child zone and any glue A records associated with the NS records. The child zone is always the authoritative zone for the delegation NS records, whereas that is not always the case for the glue A records. The inconsistency can happen either because the NS RR set declared at the parent is not the same as the one declared at the child zone, or because the glue A record does not point to the same IP address as the corresponding authoritative A record, or even because the records' TTL values are not consistent.

Delegation inconsistency is the results of coherency violation: it is caused by an inconsistency in the declaration of delegation points. For example the delegation NS RR sets, which is declared both a the parent and at the child zone, do not match, or the glue A RR and the corresponding A RR of the authoritative zone are not the same.

Delegation inconsistency errors reduce the number of all possible available authoritative servers for a zone, given that the parent and the child zone define only a subset of them, and each resolver accepts only one subset and never merges them [1]. Moreover, this kind of configuration errors can create subtle problems that are quite tricky to debug. For example inconsistencies between the glue A records at a parent and the A records defined in the child zone can create "unintentional" cache poisoning problems, meaning that DNS resolvers may cache information that are incorrect. There was a case where the delegation inconsistency led to a name server being associated with at least 13 different IP addresses [1]. Moreover the TTL inconsistencies may considerably increase the number of queries in the parent zone.

## 2.2 Lame Delegation

Lame delegation errors happen when a nameserver that is registered in the DNS system as authoritative for a zone, does not provide authoritative answers for the zone. It can be because no DNS server runs on the registered machine, or a server runs on the named machine but it does not have the authoritative data, or it replies with an error indication (Servfail or Refused error code). In all these cases we define the server as being lame and the zone as being *lame delegated*. In most times lame delegation is the result of a delegation inconsistency, where the parent zone does not correctly reflect the configuration of the child zone; in rare cases a nameserver can be lame even when the parent and the child zones have consistent delegation records but both point to a wrong server.

Lame delegation is the result of coherency violation: it is caused by an inconsistency in the declaration of authoritative nameservers. Declarations at the logical level, such as NS or A RRs, do not match with declarations at the physical level, because a nameserver is not running on the corresponding machine, or because there is a wrongly typed IP address, etc.

[1] RFC2181 [19] states that caches should never merge RR sets.

Lame delegation errors cause two major problems. First the number of actually available servers for a zone can be much lower than the one indicated by the configuration, which gives a false perception on the zone's server redundancy level. Measurements show that the majority of lame delegated zones lose as much as 50% of their servers [25]. Second, lame servers increase the response time for queries for the RRs defined in the lame zone or any other zone below. The same study shows that the response time for queries that encounter at least one lame server is one order of magnitude longer in most cases.

## 2.3 Zone Cyclic Dependency

Zone cyclic dependency errors happen when, in order to resolve a zone $Z1$, the resolver needs to query zone $Z2$, which in order to be resolved requires zone $Z1$ to be resolved first. In other words, zones $Z1$ and $Z2$ depend on each other in a mutual way. Given that most zones have multiple DNS servers, a client, in order to resolve a name, can follow multiple paths, starting from the root and going to the destination zone. Zone cyclic dependency errors occurs if there is at least one path that creates the mutual dependency. Zone cyclic dependencies errors may also involve more than two zones.

Zone cyclic dependency is the result of independence violation: it is caused by creating dependencies while declaring one or more delegation points in the logical level. For example it is likely that the availability of one server may depend on the availability of another server, under a specific configuration of NS and glue A RRs, that may involve either one or multiple zones.

Zone cyclic dependency errors can reduce a zone's reliability subsequently. Based on [25], more than 25% of the zone's servers are involved in a dependency, if the zone's configuration causes a cyclic dependency. Identifying the problem in these zones is tricky, because individual configuration files do not exhibit any apparent errors, and under normal conditions when all the servers are available, the servers that form the dependency appear to be available. Only when a combination of failures happens these problems exacerbate and these servers become unreachable, even when they are available and can provide authoritative answers for the zone.

## 2.4 Diminished Server Redundancy

Diminished server redundancy configuration errors happen when two or more servers, that are authoritative for the same zone, are placed in this way that a failure of a single component can cause the failure of these servers. For example placing the authoritative servers behind the same router can make them unavailable if the router fails[2], or placing them in the same network prefix, as being advertised by BGP, can again make them unavailable in the case of a routing problem, such as a BGP misconfiguration or route hijacking.

Diminished server redundancy is the result of independence violation: it is caused by creating dependencies while defining two or more authoritative nameservers in the physical level. For example it is possible that a zone's authoritative nameservers are placed in such a way that inability to reach one server may imply inability to reach the other servers with hight probability, given that all of them are prone to the same set of external failures.

Diminished server redundancy problems can considerably decrease the zone's availability. According to [25], zones that have all their nameservers placed behind the same /24 prefix cannot achieve even three nines of availability, while zones with nameservers placed in different networks and diverse geographic locations have at least three nines of availability. Moreover these kind of configuration errors are in general difficult to detect. The Microsoft infamous DNS problem [15] indicates that these problems can get unnoticeable for a considerable long period of time.

[2] Assuming that the router is not in a redundant configuration with another router

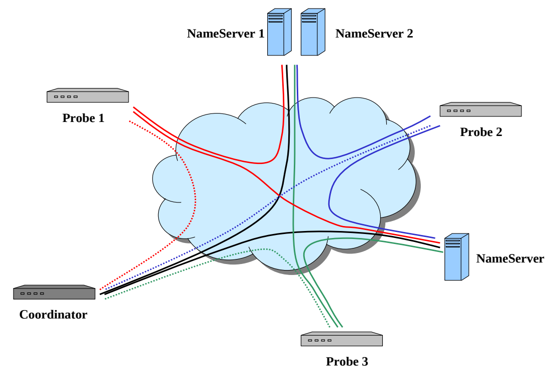


**Figure 1: An example configuration of the system: One coordinator and three probes are used in order to monitor a DNS zone with three name servers**

# 3. SYSTEM ARCHITECTURE

In this section we give a high-level description of the system architecture of our DNS troubleshooting tool. The design takes a distributed approach, due to the need of accurately identifying faults at the DNS physical level. One may be able to conduct measurements following the logical structure of the DNS hierarchy from a single vantage point. However such single point examination cannot always expose configuration errors at the DNS physical level.

For example, resolving the NS records of a zone and identifying all the zones and nameservers that can potentially be used during the resolution phase is a process that returns the same results independently of the vantage point location [3]. On the other hand, trying to identify how a single network failure may affect the reachability to the nameservers of a particular zone is a process that depends on the relative locations of the nameservers and the monitoring points. Our troubleshooting tool is designed to run at multiple vantage points located in diverse sites, and thus it is capable of exposing configuration errors at the DNS physical level.

The design defines two main components: a single coordinator and multiple probes. The coordinator has three main functionalities: *a)* performs a set of advanced diagnostic tests itself; *b)* instructs the probes to perform a set of simple diagnostic tests and collects the results, and *c)* implements the graphical user interface. The probes are DNS clients that perform simple DNS diagnostic tests, as described in section 4. The rationale behind the separation of tasks between the coordinator and the probes is the following: All the diagnostic tests related with the DNS logical level entities can be performed by a single node, and the coordinator implements this task alone to avoid unnecessary coordinations with the probes. For diagnostic tests that are related with DNS physical level entities and that require multiple vantage points, the probes implement only simple diagnostic tests and the coordinator controls the tests and process the results.

As an example, Figure 1 shows a specific configuration of a sys-

[3] There exist a few exceptions where different clients may get different answers, such as in the case of CDNs.

tem with three probes that monitor a zone which has three authoritative servers. The coordinator and the probes are installed on different machines, and the probes are distributed in diverse sites. Our design does not require the probes to be collocated with the zone's authoritative nameservers, though such collocations can be a valid deployment strategy, as we explain in section 5.

Although both the coordinator and the probes are fully capable DNS resolvers, their implementation differs from a standard DNS resolver in two fundamental ways. First, the coordinator performs a number of complex DNS querying tasks, as the one described in section 4.2.1, which are not supported by a standard resolver implementation. Second, by default the standard resolvers cache all the received RRs, a feature that needs to be disabled, because cached RR entries can conceal many DNS configuration errors.

The system queries the monitored zones periodically to identify potential configuration errors. The frequency of these queries is a trade off between promptness in detection and overhead. Ideally the frequency should be at least 2 times higher than the expected frequency of the events that we want to monitor; we believe simple empirically chosen values can provide adequate enough results for most cases. In section 4 we suggest some typical values for these frequencies.

During each poll period the following steps are performed. First, the coordinator iteratively queries the DNS system, starting from the root servers, in order to resolve the NS and glue A records of the monitored zone. This step can identify configuration errors due to the violation of the coherency or independence property at the DNS logical level only. The coordinator then instructs the probes to perform the additional diagnostic tests needed to reveal errors appearing at the DNS physical level. The results from these additional tests are sent back to the coordinator, which processes them and updates the graphical interface accordingly.

# 4. DIAGNOSTIC TESTS

In the following sections we describe how the diagnostic tests, that can identify violations of the coherency or independence property, are implemented.

## 4.1 Coherency Violation Tests

Coherency violations can happen while declaring entities either at the DNS logical or at the DNS physical level. We consider two entities: the authoritative nameserver, declared both at the logical and physical level, and the delegation point, declared only at the logical level.

### *4.1.1 Delegation Point Test*

This coherency violation diagnostic test identifies delegation inconsistency errors, and it is performed only by the coordinator. The probes are not involved in this test because, generally speaking, DNS gives the same reply to the same query, independent from where the query comes from. This is especially true for the NS and glue A records but is not always true for some other resource records, given that wide area load balancers may provide different answers to different clients [16].

The delegation inconsistency test is quite simple: the coordinator asks for the NS RR set from the zone under investigation, as well as from its parent zone. All the authoritative servers of the parent and the monitored zone are queried. In addition, for any glue A RR record defined at the parent zone, it queries the authoritative zone for that record. After collecting all the NS and glue A RR, both from the parent and the authoritative zones, the coordinator checks whether the are any inconsistencies, either at the values of these RRs or at their TTLs.

The poll period of this test is, by default, set to the minimum TTL value of all NS and glue A records of the monitored zone. The rationale behind this choice is that changes on these records are expected to happen in the frequency of their TTLs.

### *4.1.2 Authoritative Nameserver Test*

This coherency violation diagnostic test identifies lame delegation errors, and it is implemented both by the coordinator and the probes. The reason is that the same server may be reachable from some clients and unreachable from others. This can be due to transient network failures that affect only a part of the Internet, or even due to various long lived routing problems or wrong firewall configurations. It is also possible that the same server may give different replies to different clients, due to a wrongly configured DNS access list or an incorrectly defined view. The above cases indicate that lame delegation incidents can be identified better, with the coordination of many DNS clients.

The lame delegation diagnostic test for a zone is performed in the following way: each probe, as well as the coordinator, sends to each nameserver, registered as authoritative for the zone under investigation, a query for an existing resource record belonging to the zone. Then, based on the reply, it verifies if the server is indeed authoritative for the zone. The server is authoritative only if it replies with an answer that has the AA (Authoritative Answer) bit of the DNS header set. In all other cases, either there is no answer, or there is an answer with an error indication or the AA bit is not set.

There are two subtle points that we need to clarify. First this method requires a prior knowledge of a valid RR for the zone. This requirement is fulfilled with the use of the zone's SOA (Start of Authority) RR, which is the only record that is always defined if the zone exists [4]. Secondly there is a thin line in the definition of a lame server and a server that is temporary unavailable, due to transient routing problems. Thus, in the case that a server is not responding, we consider it as being lame if that happens continuously for a long period of time [5]. The poll period of the lame delegation diagnostic test is configurable, and its default value is set to the TTL value of the NS RR.

## 4.2 Independence Violation Tests

As in the case of coherency violations, independence violations can happen while defining entities either at the DNS logical or at the DNS physical level. Again, we consider the same two entities: the authoritative nameserver and the delegation point.

### *4.2.1 Delegation Point Test*

This independence violation diagnostic test identifies cyclic dependency errors, and it is performed only by the coordinator. Again, because all the potential DNS logical paths, that a client may follow in order to resolve one zone, are independent from its location, this test is performed by the coordinator only.

The coordinator first identifies all the potential paths, by iteratively querying the DNS system and by visiting all the identified servers. The algorithm follows a breadth first approach, meaning that the coordinator visits all the servers that are defined in each NS RRs and then proceeds to the next level of the DNS hierarchy. For NS records that their IP address needs to be resolved, i.e. when there is no glue A record, the resolver starts a new instantiation of the algorithm, and temporarily holds the previous instance in a

[4] Note that NS records are also required for a correctly defined zone.

[5] RFC2308 defines a nameserver as dead if it doesn't respond within 2 minutes.

pending stage. A cyclic dependency is detected if during this procedure a pending instance needs to be prematurely resumed. When this happens, all the zones whose instances are in a pending stage are involved in the cyclic dependency, as well as all the servers that form the loop.

The poll period for this type of tests is set to the minimum TTL value of the NS records of all involved zones, again for the same reason as previously described.

#### 4.2.2 Authoritative Nameserver Test

This independence violation diagnostic test identifies diminished server redundancy errors, and it is performed by both the coordinator and the probes. The reason is that different kinds of component failures can affect different parts of the network. Therefore, observations from multiple points can increase the confidence of the reported results.

More specifically, the diminished server redundancy test is performed as follow: each probe, as well as the coordinator, sends an SOA query to all the authoritative servers of the zone and waits for a response. In the case it does not receive a response, after trying a number of times, it performs a *traceroute* like probe in order to identify the last router that is capable of forwarding the query. The coordinator receives these unreachability information, and based on the reports coming from multiple probes, can estimate how severe is a failure and also the type of the failure. Routing failures appear to have very diverse last forwarding routers, whereas single box failures give the same last forwarding router, in most cases.

The polling period of this test highly depends on the time duration of the transient network failures; it can range from multiple minutes to several hours. Zone operators may specify a proper frequency for their zones.

## 5. SYSTEM DETAILS

In this section we present some more detailed aspects of the system. We briefly explain how it is implemented and how it can be configured. Finally, we demonstrate the main features of the user interface.

### 5.1 Implementation

Both the coordinator and the probes are implemented in the Perl scripting language. This decision allowed us to use a great number of libraries. For example we make use of the NetDNS [9] module in order to construct the DNS packets. In addition we use the GraphViz [6] and RRDTool [10] modules in order to create the user interface graphs.

### 5.2 Configuration

In order to make the configuration of the whole system as simple as possible, operators are required to configure only the following parameters: the domain names of the monitored zones, and the IP addresses of the probes. Optionally, other parameters, such as frequency of the diagnostic tests or the output type of the results, can be configured.

While the coordinator is usually installed on a machine which is under the administration of the zone's operator, the probes can be installed on machines that may be under different administration. It is worthwhile to discuss the possible options that an operator may have in order to select the monitoring machines. In the simplest case the probes can be collocated with the zone's nameservers. Another option is that the probes are selected from a big pool of machines that publicly allow simple DNS queries, similar to the case of public available traceroute servers or looking glasses. Finally, servers that allow arbitrary types of network measurements [26] can be a third option.

### 5.3 User Interface

Our DNS troubleshooting tool also implements a comprehensive user interface. The interface is web-based: the coordinator creates a number of HTML pages and graphs that are served by a standard web server and can be viewed through a web browser. The user interface provides two types of graphs: a configuration graph that shows DNS entities of the logical and physical level. It includes only the entities that belong to the zones that may potentially be used in order to resolve the monitored zone. A second type of graph is the statistics graph that shows the history of events, captured during the last day, week or month.

Figure 2 shows the combined logical and physical level graph for a zone. It actually shows that the *unisourse.it* zone is in a cyclic dependency with the *ita.tip.net* zone. The configuration of two authoritative servers, the *ns.unisourse.it* and the *ns2.ita.tip.net*, creates the problem. Note that zones are presented with rectangles and servers are presented with oval shapes. Zones that experience at least one configuration error are marked with red color, whereas servers that appear to have at least one configuration error are marked with yellow color. Additional information, such the type of the configuration error, is displayed above the misconfigured servers or zones.

## 6. RELATED WORK

There are a few tools that can identifying DNS misconfigurations [11, 3, 5, 4, 2]. Most of them can detect only a subset of the DNS infrastructure related configuration errors. On the other hand, they implement diagnostic tests that can identify errors related with application specific RRs, such as MX RR or PTR RR. And, to the best of our knowledge, they are limited to one monitoring machine, and thus they may not be able to detect a number of configuration errors that require distributed diagnosis.

There exist an SNMP MIB defined for the DNS operations [13], but it is mostly oriented for accounting purposes, and thus it is relative hard to make use of this MIB in order to identify configuration problems. Moreover, complex network management tools, such as HP's OpenView [7] and IBM's Tivoli [8], can be configured or programmed in order to diagnose network related problems. On the other hand, these tools heavily rely on SNMP MIBs and thus they are not adequate to address most of the problems described in this paper.

Duane [27] classifies different types of bogus DNS queries, that are commonly seen by the DNS root servers, and he describes methods of identifying misbehaving DNS resolvers and ways of fixing these errors. Finally, in a related work [25] we present and classify the DNS configurations errors described in this paper, and measure their pervasiveness and impact on the global DNS system.

## 7. SUMMARY

DNS is a manually configured distributed database which crosses multiple administrative domains. It is a well known fact that configuration errors not only exist but are also widespread. To facilitate operators with DNS misconfiguration detection we have developed a troubleshooting tool which identifies configuration errors that can affect the DNS infrastructure. Our design takes the approach of first identifying basic properties of a correct configuration and then detecting violations of those properties. We also utilize multiple monitoring points to diagnose misconfigurations that cannot be easily detected from a single point.

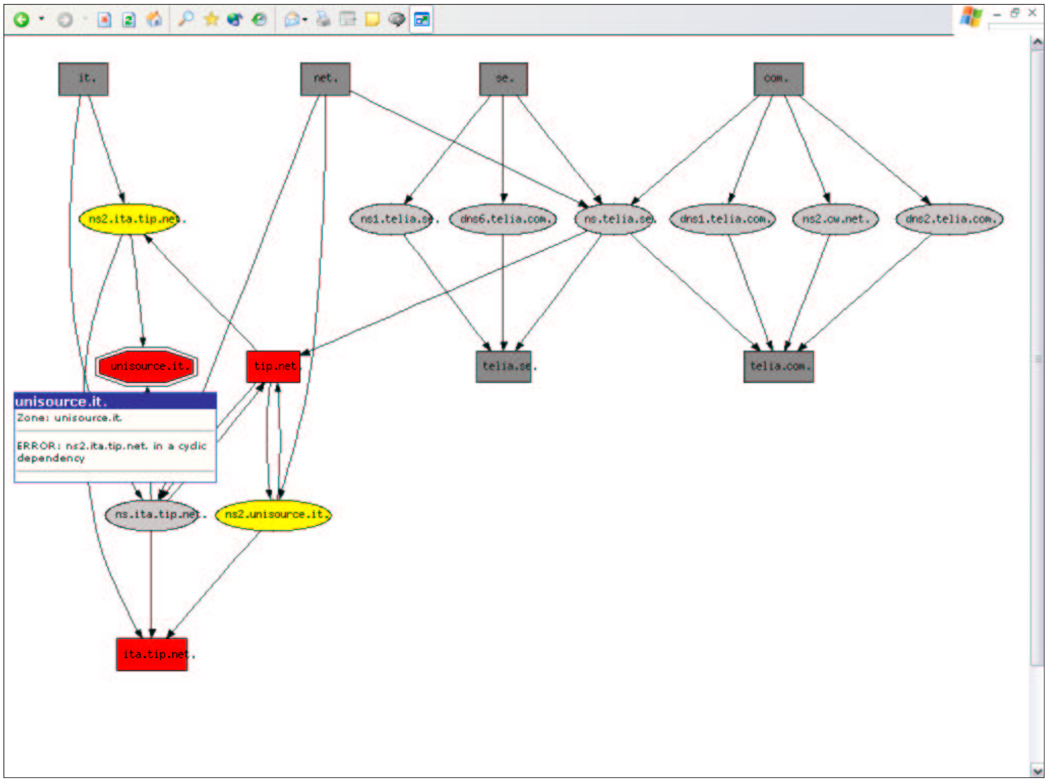


**Figure 2: A snapshot of the user interface: a cyclic dependency error affecting the *unisourse.it* zone**

As a future work, we plan to extent our tool to detect configuration errors associated with the DNSSEC deployment. At the same time we seek to sharpen our understanding of the basic properties of correctly configured systems, in order to detect both known and unknown configuration errors.

## 8. REFERENCES

[1] Private communication with Mark Andrews, Nov. 2003.
[2] CheckDNS. http://www.checkdns.net/quickcheck.aspx.
[3] DNSCheck. http://dnscheck.se/.
[4] DNSChecker. http://www.squish.net/dnscheck/.
[5] DNSReport. http://www.dnsreport.com/.
[6] GraphViz. http://www.research.att.com/sw/tools/graphviz/.
[7] HP OpenView. http://www.openview.hp.com/.
[8] IBM Tivoli. http://www.ibm.com/software/tivoli/.
[9] Net::DNS. http://www.net-dns.org/.
[10] RRDTool. http://people.ee.ethz.ch/ oetiker/webtools/rrdtool/.
[11] ZoneCheck. http://www.zonecheck.fr/.
[12] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. Work in progress: draft-ietf-dnsext-dnssec-intro-08, December 2003.
[13] R. Austein and J. Saperia. DNS Server MIB Extensions. RFC 1611, 1994.
[14] D. Barr. Common DNS Operational and Configuration Errors. RFC 1912, 1996.
[15] N. Brownlee, k claffy, and E. Nemeth. DNS Measurements at a Root Server. In *Proceedings of the IEEE Globecom' 01*, pages 1672–1676, 2001.
[16] Cisco Systems. Distributed Director. http://www.cisco.com/.
[17] Credentia. http://www.credentia.cc/research/cctlds/, 2004.
[18] Team Cymru. http://www.cymru.com/DNS/lame.html, 2004.
[19] R. Elz and R. Bush. Clarifications to the DNS Specification. RFC 2181, 1997.
[20] R. Elz, R. Bush, S. Bradner, and M. Patton. RFC 2182 - selection and operation of secondary dns servers. Rfc, 1997.
[21] Ed Lewis. Implementation of ARIN's Lame DNS Delegation Policy. http://www.nanog.org/mtg-0306/lewis.html, 2003.
[22] Men & Mice. http://www.menandmice.com/, 2004.
[23] P. Mockapetris. Domain Names–Concepts and Facilities. RFC 1034, 1987.
[24] P. Mockapetris. Domain Names–Implementation and Specification. RFC 1035, 1987.
[25] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang. Impact of Configuration Errors on DNS Robustness. In *Proceedings of the ACM SIGCOMM'04*, 2004.
[26] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A Public Internet Measurement Facility. In *Proceedings of the Usenix Symposium on Internet Technologies and Systems*, 2003.
[27] D. Wessels. Is Your Caching Resolver Polluting the Internet. In *Proceedings of the ACM SIGCOMM workshop on Network Troubleshooting*, 2004.