

# Issue Review: DNS Error Caused Email Bounces

Technology Architecture Group

March 21, 2008

[www.oit.duke.edu/tag](http://www.oit.duke.edu/tag)

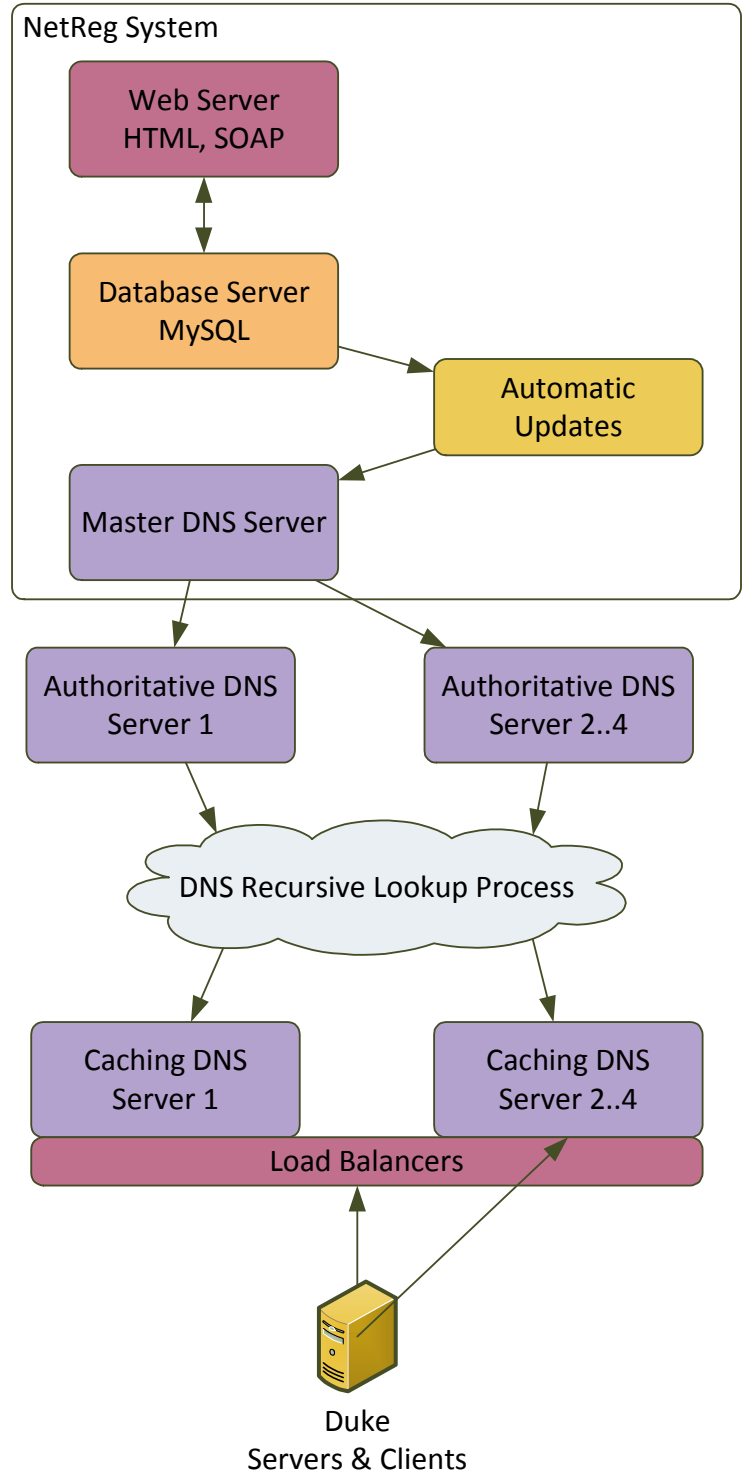
*Issue Reviews are intended to provide insight into the technical "lessons learned" from real decisions and issues faced by OIT staff. We hope they are helpful in thinking about the application of the Architectural Principles to your work.*

On the afternoon of Wednesday, March 12, an OIT staff member within the Network Services group was troubleshooting a problem establishing a new third level domain (*cbcb.duke.edu*). Though unknown until after the incident, the source of the problem with the domain was an automatic software update to the DNS server package that had unknowingly changed the permissions on a data directory.

As the routine troubleshooting proceeded, an attempt was made to remove the *localhost.cbcb.duke.edu* entry from the NetReg database. In doing so, the *localhost* entries for other domains were inadvertently removed from the database. (The *localhost* entries are linked by necessity.) Regular, automatic sweeps of the NetReg database identify pending changes to DNS and send them to the Master DNS Server, which quickly propagates them to the Authoritative DNS servers.

***Otherwise routine troubleshooting led to the inadvertent removal of 'localhost' entries.***

Servers and client workstations at Duke query a separate set of Caching DNS servers. These servers query Duke's authoritative DNS servers and other authoritative servers around the world to provide DNS answers. Answers are cached and therefore all servers did not immediately observe the loss of the *localhost* entries.



Although the now-missing *localhost* entries were restored to the database approximately 30 minutes after the removal, the problems were just beginning. One caching server had observed the erroneous information and began providing this information to other systems, including Duke's main email system. This system did not have a standalone *localhost* entry and without an affirmative *localhost* answer, it began bouncing incoming email to Duke recipients.

Though the caching DNS servers became accessible through a load balancer in 2007, most systems operate with DNS server configurations that point at a single caching DNS server. This was the case for the Duke email system.

Unfortunately, the single server was the one that cached the erroneous *localhost* information.

The absence of any information for *localhost.oit.duke.edu* would, by the DNS server's default configuration, have been "negatively" cached for 3 hours after it was retrieved.

The email delivery problems were noted by the Help Desk shortly after the *localhost* entries were restored in the database and authoritative servers. The link between issues was not immediately clear, and diagnosis of the email problem proceeded independently until the link was established.

When the *localhost* information was identified as the source of the problem, the invalid information was manually flushed from the caching DNS server. The server subsequently re-queried the authoritative server and began answering with the proper information. This restored the email delivery service.

## ***TAG Recommendations***

1. Shorten default negative cache timers  
*Principle: Create robust, secure systems*  
Shorter cache times would enable swifter recovery to expected values in this highly automated system.
2. Provide an "undo" feature in NetReg  
*Principle: Adapt to resource realities*  
Human errors are inevitable; provide mechanisms to recover quickly.
3. Provide a "view recent changes" feature in NetReg  
*Principle: Design for Info Lifecycles*  
Fixing inter-system errors could be aided by knowledge of recent changes in data.
4. Provide a feature to lock critical records in NetReg  
*Principle: Adapt to resource realities*  
Human errors are inevitable; automate sanity checks as much as possible.
5. Ensure systems are utilizing load balanced DNS servers  
*Principle: Design for scalability*  
The load balancer provides horizontal scaling capability without requiring client-side configuration.
6. Establish end-to-end functional testing of email system  
*Principle: Create robust, secure systems*  
Automate functional testing of the email system as a whole to provide rapid notification of failures.