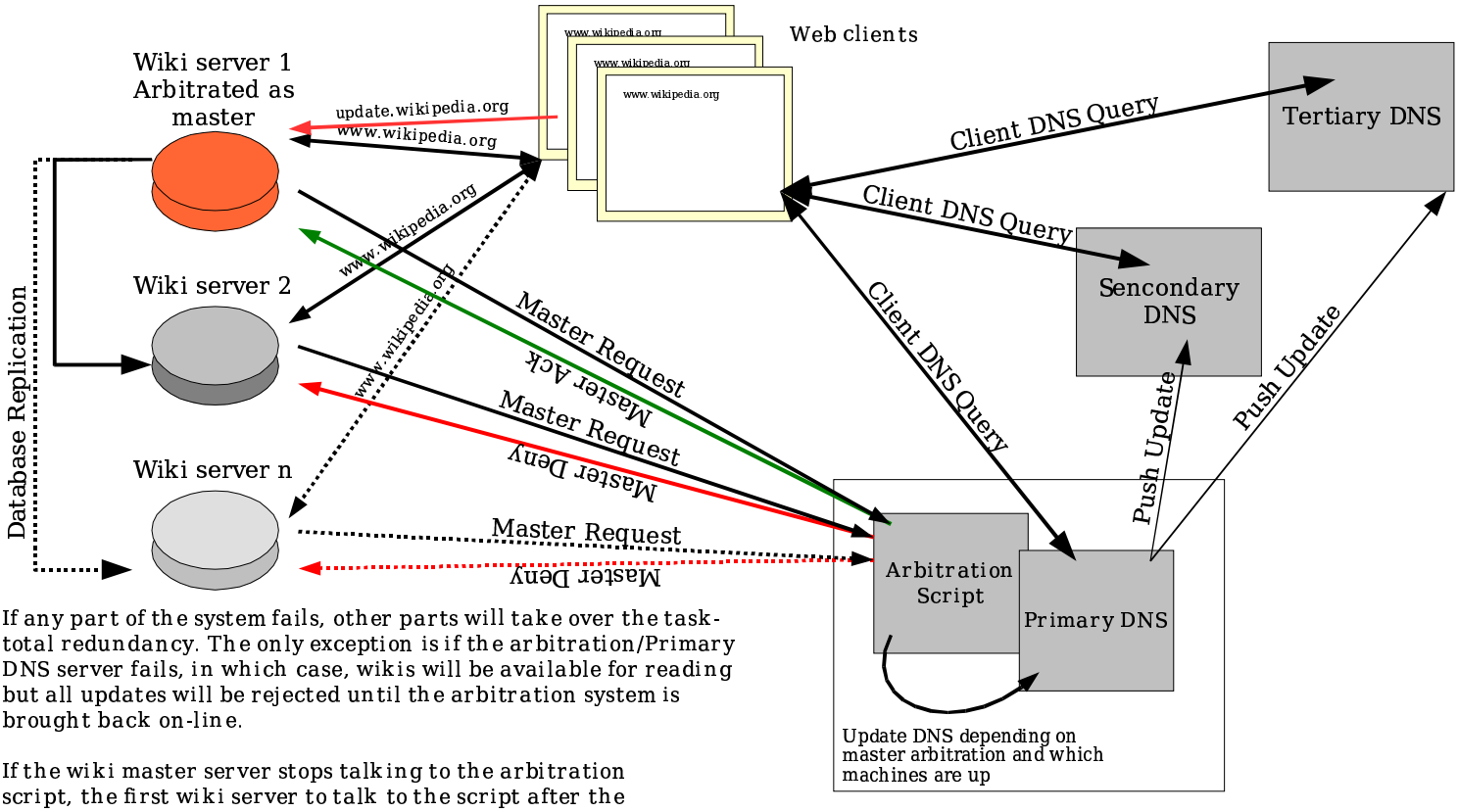
Suggestion for redundant wiki server configuration

2nd Jan 2004



If any part of the system fails, other parts will take over the task- total redundancy. The only exception is if the arbitration/Primary DNS server fails, in which case, wikis will be available for reading but all updates will be rejected until the arbitration system is brought back on-line.

If the wiki master server stops talking to the arbitration script, the first wiki server to talk to the script after the time-out will be elected as master. The DNS will immediately be updated.

If a wiki server stops talking to the arbitration script, it's entry will be removed from the round robin DNS and will no longer be sent queries by web clients. Once it starts talking to the arbitration script again, it's entries are automatically restored and re-joins the team.

All servers will take equal load for normal queries. Only the master as elected and represented in the DNS will accept updates. Load can be apportioned by using multiple round robin entries.

All servers are of the same initial configuration except for ip address and MySQL database id. Wiki servers re-configure themselves if elected or de-selected as master. Slave servers re-configure themselves based on which server is currently master, as defined by the DNS entry. Ideally, each wiki is on a different network to avoid any single points of failure.

When a user types in www.wikipedia.org into his browser, the resolver on his machine will contact one of the DNS servers of wikipedia.org for an ip address of a machine which can fill his request. We make the system reliable by controlling which ip address he is given. The system ensures the ip address he receives is always that of a working server.

The arbitration script, located on the primary DNS, enters the ip address of all capable servers into the DNS system. As far as the user is concerned, he will receive an ip of a suitable server picked at random. (it isn't really random but for our purposes it is).

As MySQL database replication is unidirectional, only one server can accept updates. We therefore need a system to elect one server as a master. In order to avoid single points of failure, all servers should be capable of switching to master mode and back again. We will therefore assume all servers are equal and can be switched from master to slave mode and back againthrough the action of a script.

The entries in the DNS server and therefore the possible ip addresses which will be handed out to clients is controlled by a simple arbitration script. The arbitration script is invoked by the wikipedia servers at timed intervals. The arbitration script changes the host file and forces a BIND reload/update whenever a server comes on-line or fails to report after a defined time period. If the master fails to report, a new master is elected - the ip address of the hostname associated with the server able to accept updates is then changed. The exact algorithm the script uses is outside the scope of this document but has been documented in my post to the wikitech-l email group. It is simple and can be implemented using Bash scripting and/or PERL.

If the master wiki server fails to receive an acknowledgement (ack) it must immediately demote itself to slave. If the current master fails to invoke the arbitration script for a pre-defined period, the next machine to make a request is granted master status. DNS records are immediately updated to point to the newly elected master.

Each time a wiki server receives a DENY from the arbitration server, the script on the wiki server should check that it's own record of the master database ip corresponds to the DNS record, then check Mysql is configured as a slave server, changing the configuration as required. If config changes, issue a MySQL CHANGE MASTER TO command with a zero binary log.

Whenever a wiki server receives an ACK to a master request, it must check it is set as a master else re-configure itself as a master. This may first require a MySQL RESET MASTER to clear any stale binary logs.

IMPLEMENTATION NOTES:

The Wiki servers invoke their script and hence the arbitration script through cron. The script first checks the local database and web server are operating correctly by fetching a database page and comparing this with the expected result. If the local system is functioning, the script continues by connecting to the arbitration server via SSH. The system is secure, using encrypted connections and public/private keying. Also, ip-based authentication. The arbitration script return value (master ack/deny) is communicated to the wiki server script over the SSH transport. Depending on the return value, the local arbitration script may make local configuration changes to: The status of the machine as master/slave, if slave, verify/change the address of the master server, issue MySQL commands to change master or reset master.

The arbitration script may directly modify the /var/cache/bind/wikipedia.org.hosts file (the arbitration script is located on the primary DNS server) and force the local BIND DNS system to reload. This, in turn, pushes updates to the secondary DNS servers. The DNS TTL is set to a low value so that ip address changes propogate within a few minutes.

If the wiki server script fails to verify the functionality of the local apache/MySQL system, the script will not talk to the arbitration server. The arbitration server will respond by removing the wiki server from the list of ip addresses. Clients will stop sending queries to that machine within minutes.

The system does not need any unusual software. Only Bash/PERL, cron, Bind (v8 or later) and SSH are needed.