DNS Cache Monitor

1 Introduction

The cache monitor is running on every instance of an anycast DNS cache. It performs local checks to determine whether the cache is working properly. Should any of these checks fail, it attempts to restore the cache to an operational state. Repeated failures cause the cache to be taken offline for a holddown period which is increasing exponentially up to a pre-set maximum if the problem is persistent.

The monitoring process takes as input a set of `<anycast address, interface>` pairs and a set of domain name queries in the `IN` class specified as `<domain name, query type>` pairs.

```
<address_1, iface_1>
.
.
<address_n, iface_n>
<qname_1, qtype_1>
.
.
<qname_m, qtype_m>
```

For example:

```
<130.59.31.251, lo:1>
<2001:620:0:FF::2, lo>
<cache-mon-1.net.switch.ch., TXT>
<cache-mon-2.net.switch.ch., TXT>
<cache-mon-3.net.switch.ch., TXT>
```

Each DNS resource record referenced by these queries should have a TTL of 0 and they should all be served by physically independent name servers. The first condition gurantees that the "DNS cache availability check" described below will always require the cache to perform queries to remote hosts rather than answering the query from its cache. The second condition makes the check robust against the failure of any of the name servers for the zones that contain these resource records.

For the full set of command-line options, see the man page.

2 Operation

2.1 Configuration

The process can be configured by editing the file `/etc/default/dns-cache-monitor` and restarting the daemon. This file should set the shell variables `DAEMON_OPTS`, `ADDRV4` and `ADDRV6`. `DAEMON_OPTS` can contain any of the valid daemon options, while `ADDRV4` and `ADDRV6` should contain exactly one address specification for the corresponding address family as described in the man page. The address specifications should not be edited by hand. The preferred method to manipulate them is by issuing the command

```
bash# dpkg-reconfigure dns-cache-monitor
```

A typical configuration looks like this:

```
DAEMON_OPTS="--daemonize --mailto=noc@switch.ch"
ADDRV4="130.59.31.248/lo:1"
ADDRV6="2001:620:0:FF::3/lo"
```

2.2 Start/Stop

The daemon is crontrolled through an init script in the usual manner, i.e.

```
bash#/etc/init.d/dns-cache-monitor start
```

to start,

```
bash#/etc/init.d/dns-cache-monitor stop
```

to stop and

```
bash#/etc/init.d/dns-cache-monitor restart
```

to restart. When the daemon has detected an error condition, it sleeps for some time before making the next attempt to activate the service. This holddown period can be as long as 2 hours. The daemon can be forced to end the holddown immediately with

```
bash#/etc/init.d/dns-cache-monitor resume
```

Note that when the cache monitor is stopped, the name server is not automatically stopped as well.

2.3 Logging and notification

All events and actions are written to the log file (refer to the man page for details where the information is written to). Whenever a state transition occurs, an email is sent to the address specified by the `–mailto` option of `dns-cache-monitor`, including the last 15 log messages.

3 State Machine

The monitoring process is implemented as a finite state machine (FSM). This section describes the events, states, actions and transitions between states that make up the FSM.

3.1 Events

The `dns-cache-monitor` periodically checks three conditions (every 4 seconds by default). Each condition can be either true or false and each outcome represents an event that drives the FSM.

3.1.1 Interfaces up (E_IFUP)

Each interface is checked to be marked as `UP` by the kernel according to the pseudo-code

```
E_IFUP = true
foreach interface {
  if not up(interface) {
    E_IFUP = false
  }
}
```

3.1.2 Local addresses reachable (E_REACH)

For each interface that is up, it is verified whether the associated anycast address is locally reachable with `ICMP` echo requests (the TTL of the echo request packets is limited to 1 to make sure they don't leave the host). The result for each interface is stored for use in the last event (`E_CA`).

```
E_REACH = true
if (E_IFUP) {
  foreach address {
    if not ping_with_ttl_1(address) {
      E_REACH = false
      address.reach = false
    }
  }
} else {
  E_REACH = false
}
```

3.1.3 DNS cache available (E_CA)

The local DNS cache must be configured to listen on all anycast addresses as well as the loopback addresses `127.0.0.1` and `::1`. For each anycast address that has been checked to be reachable, it is verified whether the cache can iteratively resolve DNS queries sent to this address. If an address is not marked as reachable, the loopback address for the corresponding address family is used instead.

First, the list of queries supplied as input to the monitoring process is sorted in an arbitrary order. Next, each query is tried in turn until either one succeeds or the list is exhausted.

If at least one query can be resolved sucessfully for all addresses or the loopback addresses, the cache is marked to be available.

```
E_CA = true
foreach address {
  query_address = address
  if not address.reach {
    switch(address_family(address)) {
      case ipv4:
        query_address = 127.0.0.1
      case ipv6:
        query_address = ::1
    }
  }
  randomize_queries()
  foreach query {
    reply = dns_query(query_address, query.name, query.type)
    if not (reply.rcode == NOERROR and reply.answers == 1) {
      E_CA = false
    }
  }
}
```

3.2 States

3.2.1 S_INIT

The state S_INIT is entered when the process starts up. After initialization, the FSM transitions unconditionally to S_UP and never enters S_INIT again.

3.2.2 S_UP

When in this state, all anycast addresses are reachable and the Cache is working properly on all of them. This is represented by the following combination of events

```
S_UP: E_CA && E_REACH
```

3.2.3 S_IDOWN

In this state, at least one interface holding one of the anycast addresses is down but the cache is working properly at least on the loopback addresses (note that !E_IFUP implies !E_REACH).

```
S_IDOWN: E_CA && !E_IFUP
```

3.2.4 S_DOWN

In this state, the cache is not working for at least one address family and at least one interface is down.

```
S_DOWN: !E_CA && !E_IFUP
```

3.3 Actions

The actions described below are executed during state transitions.

3.3.1 A_IFUP

Execute the system commands to bring all interfaces up.

3.3.2 A_IFDOWN

Execute the system commands to bring all interfaces down.

3.3.3 A_RCFG

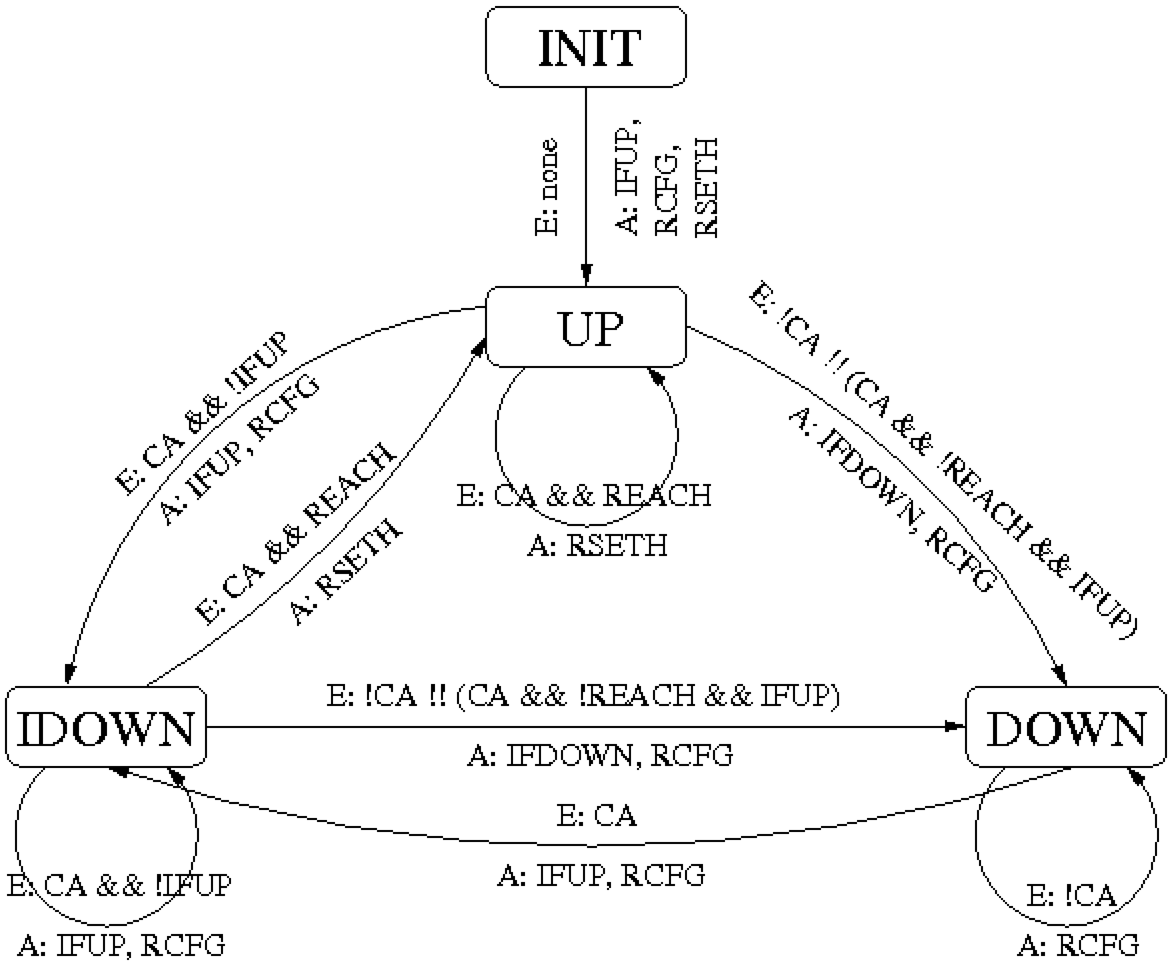
If the cache is still running, signal it to reconfigure itself. This is required at least for BIND to listen or stop listening on interfaces that have come up or gone done, respectively.

If the cache is not running, it is started.

3.3.4 A_RSETH

Reset the holddown timer to its initial state.

3.4 State transitions



Appendix

A Debian Package

The software is available as a Debain package called `dns-cache-monitor`. To build an architecture-independent package, check out the source [http://cvs.switch.ch/cgi-bin/viewcvs.cgi/dns-cache-monitor/?root=network] from CVS

```
bash$ CVSROOT=:ext:cvs.switch.ch:/reps/cvs/network cvs co dns-cache-monitor
```

and execute `dpkg-buildpackage -rfakeroot` in the CVS sandbox. To make a new release, edit `debian/changelog` accordingly before building the package. Each release should be tagged with

```
bash$ cvs tag -c release_<major>-<minor>_<revision>
```

for example

```
bash$ cvs tag -c release_0-4_1
```

B Loopback interfaces and routing issues

The Quagga routing suite is a prerequisite dependency for the `dns-cache-monitor` package. It is required to inject the anycast addresses into the internal routing system. See the documentation of the BGP configuration for host-routes for details.

When the package is first installed, an unused sub-interface of the loopback interface is selected for each anycast address of the local DNS cache instance.

Operating systems differ in the manner in which IPv4 and IPv6 addresses can be configured on the loopback interface.

Solaris

On Solaris, every physical interface (the loopback device is considered to be physical as well in this context) can be configured with an arbitrary number of IPv4 and IPv6 addresses. Each address is assigned to a separate sub-interface (also called logical interface) of the corresponding physical interface, e.g.

```
: gall@tahoma[gall]; /sbin/ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
lo0:1: flags=1000849 mtu 8232 index 1
        inet 130.59.31.8 netmask ffffffff
lo0:2: flags=1000849 mtu 8232 index 1
        inet 130.59.211.10 netmask ffffffff
bge0: flags=1000843 mtu 1500 index 2
        inet 130.59.138.22 netmask ffffff00 broadcast 130.59.138.255
bge1: flags=1000843 mtu 1500 index 3
        inet 130.59.138.23 netmask ffffff00 broadcast 130.59.138.255
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
lo0:1: flags=2000849 mtu 8252 index 1
        inet6 2001:620::8/128
lo0:2: flags=2000849 mtu 8252 index 1
        inet6 2001:620::5/128
bge0: flags=2000841 mtu 1500 index 2
        inet6 fe80::203:baff:feaa:b28f/10
bge0:1: flags=2080841 mtu 1500 index 2
        inet6 2001:620:0:1b:203:baff:feaa:b28f/64
bge1: flags=2000841 mtu 1500 index 3
        inet6 fe80::203:baff:feaa:b290/10
bge1:1: flags=2080841 mtu 1500 index 3
        inet6 2001:620:0:1b:203:baff:feaa:b290/64
```

This is very convenient, because an address can be activated or deactivated by simply configuring the corresponding sub-interface as up or down, respectively.

Linux

On Linux, additional IPv4 addresses can only be configured on sub-interfaces, while IPv6 addresses can only be configured on the physical interface, e.g.

```
: gall@yasur[gall]; ifconfig lo
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          inet6 addr: 2001:620:0:ff::2/128 Scope:Global
          inet6 addr: 2001:620::9/128 Scope:Global
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:9495122147 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9495122147 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:13746229324572 (12.5 TiB)  TX bytes:13746229324572 (12.5 TiB)

: gall@yasur[gall]; ifconfig lo:0
lo:0      Link encap:Local Loopback
          inet addr:130.59.31.9  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:16436  Metric:1

: gall@yasur[gall]; ifconfig lo:2
lo:2      Link encap:Local Loopback
          inet addr:130.59.31.70  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

This is very inconvenient for IPv6, because in order to activate or deactivate an address, it must be added or removed from the `lo` interface.

network/docu/dns/dns_cache_monitor.txt · Last modified: 2010/04/13 15:26 by 2001:620:0:4:203:baff:fe4c:d99b