Scaling IP address handling in CTDB

Martin Schwenke <martin@meltin.net>

Samba Team
IBM (Australia Development Laboratory, Linux Technology Center)

Public IP addresses

- CTDB uses a pool of IP addresses to provide high availability and (weak) load balancing

Public IP addresses

- CTDB uses a pool of IP addresses to provide high availability and (weak) load balancing
- When a node is hosting public IP addresses and it becomes **unhealthy** then the IP addresses are redistributed to other **healthy** nodes

Public IP addresses

- CTDB uses a pool of IP addresses to provide high availability and (weak) load balancing
- When a node is hosting public IP addresses and it becomes **unhealthy** then the IP addresses are redistributed to other **healthy** nodes
  1. **releaseip:** Each node *releases* each public IP address that it should not be hosting

Public IP addresses

- CTDB uses a pool of IP addresses to provide high availability and (weak) load balancing
- When a node is hosting public IP addresses and it becomes **unhealthy** then the IP addresses are redistributed to other **healthy** nodes
  1. **releaseip:** Each node *releases* each public IP address that it should not be hosting
  2. **takeip:** Each node *takes* each public IP address that it should be hosting

Public IP addresses

- CTDB uses a pool of IP addresses to provide high availability and (weak) load balancing
- When a node is hosting public IP addresses and it becomes **unhealthy** then the IP addresses are redistributed to other **healthy** nodes
  1. **releaseip:** Each node *releases* each public IP address that it should not be hosting
  2. **takeip:** Each node *takes* each public IP address that it should be hosting
  3. **ipreallocated:** Each node reconfigures (network, NAS, ...) services that depend on the allocation of public IP addresses

Public IP addresses

- CTDB uses a pool of IP addresses to provide high availability and (weak) load balancing
- When a node is hosting public IP addresses and it becomes **unhealthy** then the IP addresses are redistributed to other **healthy** nodes
  1. **releaseip:** Each node *releases* each public IP address that it should not be hosting
  2. **takeip:** Each node *takes* each public IP address that it should be hosting
  3. **ipreallocated:** Each node reconfigures (network, NAS, ...) services that depend on the allocation of public IP addresses
- Alternative approaches include LVS

takeip and **releaseip** events

- CTDB uses “event scripts” to manipulate public IP addresses and manage services

takeip and **releaseip** events

- CTDB uses "event scripts" to manipulate public IP addresses and manage services
- The event scripts contain **takeip** and **releaseip** (and **updateip**) events for manipulating IP addresses

takeip and **releaseip** events

Taking an IP...

```
case "$1" in
  takeip)
    iface=$2
    ip=$3
    maskbits=$4

    add_ip_to_iface $iface $ip $maskbits ||
        exit 1;


    # cope with the script being killed while we have the interface blocked
    iptables -D INPUT -i $iface -d $ip -j DROP 2> /dev/null

    # flush our route cache
    set_proc sys/net/ipv4/route/flush 1
    ;;
```

takeip and **releaseip** events

15 scripts are enabled by default...

```
[root@m1n1 ~]# ctdb scriptstatus
15 scripts were executed last monitor cycle
00.ctdb              Status:OK    Duration:0.012 Tue May  6 10:47:24 2014
01.reclock           Status:OK    Duration:0.016 Tue May  6 10:47:25 2014
10.interface         Status:OK    Duration:0.033 Tue May  6 10:47:25 2014
11.natgw             Status:OK    Duration:0.016 Tue May  6 10:47:25 2014
11.routing           Status:OK    Duration:0.011 Tue May  6 10:47:25 2014
13.per_ip_routing    Status:OK    Duration:0.015 Tue May  6 10:47:25 2014
20.multipathd        Status:DISABLED
31.clamd             Status:DISABLED
40.fs_use            Status:DISABLED
40.vsftpd            Status:OK    Duration:0.021 Tue May  6 10:47:25 2014
41.httpd             Status:OK    Duration:0.013 Tue May  6 10:47:25 2014
49.winbind           Status:OK    Duration:0.011 Tue May  6 10:47:25 2014
50.samba             Status:OK    Duration:0.045 Tue May  6 10:47:25 2014
60.ganesha           Status:OK    Duration:0.013 Tue May  6 10:47:25 2014
60.nfs               Status:OK    Duration:0.238 Tue May  6 10:47:25 2014
62.cnfs              Status:OK    Duration:0.011 Tue May  6 10:47:25 2014
70.iscsi             Status:OK    Duration:0.010 Tue May  6 10:47:25 2014
91.lvs               Status:OK    Duration:0.009 Tue May  6 10:47:25 2014
99.timeout           Status:DISABLED
```

takeip and **releaseip** events

The problem

- **takeip** and **releaseip** event each run in parallel
- Some users have lots of public IP addresses

takeip and **releaseip** events

The problem

- **takeip** and **releaseip** event each run in parallel
- Some users have lots of public IP addresses
  - 10s, 100s, ...

takeip and **releaseip** events

The problem

- **takeip** and **releaseip** event each run in parallel
- Some users have lots of public IP addresses
  - 10s, 100s, ...
  - Large number of nodes, fine grained balancing

takeip and **releaseip** events

The problem

- **takeip** and **releaseip** event each run in parallel
- Some users have lots of public IP addresses
  - 10s, 100s, ...
  - Large number of nodes, fine grained balancing
  - Multiple networks/VLANs

takeip and **releaseip** events

The problem

- **takeip** and **releaseip** event each run in parallel
- Some users have lots of public IP addresses
  - 10s, 100s, ...
  - Large number of nodes, fine grained balancing
  - Multiple networks/VLANs
- When a lot of **takeip** and **releaseip** events run at once then this can generate a high load and take a long time

takeip and **releaseip** events

The problem

- **takeip** and **releaseip** event each run in parallel
- Some users have lots of public IP addresses
  - 10s, 100s, ...
  - Large number of nodes, fine grained balancing
  - Multiple networks/VLANs
- When a lot of **takeip** and **releaseip** events run at once then this can generate a high load and take a long time
- IP failover can time out

takeip and **releaseip** events

The problem

- **takeip** and **releaseip** event each run in parallel
- Some users have lots of public IP addresses
  - 10s, 100s, ...
  - Large number of nodes, fine grained balancing
  - Multiple networks/VLANs
- When a lot of **takeip** and **releaseip** events run at once then this can generate a high load and take a long time
- IP failover can time out
- Nodes can be banned

takeip and **releaseip** events

The problem

- **takeip** and **releaseip** event each run in parallel
- Some users have lots of public IP addresses
  - 10s, 100s, ...
  - Large number of nodes, fine grained balancing
  - Multiple networks/VLANs
- When a lot of **takeip** and **releaseip** events run at once then this can generate a high load and take a long time
- IP failover can time out
- Nodes can be banned
- ...and that's a problem!

takeip and **releaseip** events

Solution #1: Batch **takeip** and **releaseip**

takeip and **releaseip** events

Solution #1: Batch **takeip** and **releaseip**

- Add a configuration tunable so that **TAKEIP** and **RELEASEIP** *controls* get queued — so no scripts are run

takeip and **releaseip** events

Solution #1: Batch **takeip** and **releaseip**

- Add a configuration tunable so that **TAKEIP** and **RELEASEIP** *controls* get queued — so no scripts are run
- Add new controls: **TAKEIPBATCH** and **RELEASEIPBATCH**

takeip and **releaseip** events

Solution #1: Batch **takeip** and **releaseip**

- Add a configuration tunable so that **TAKEIP** and **RELEASEIP** *controls* get queued — so no scripts are run
- Add new controls: **TAKEIPBATCH** and **RELEASEIPBATCH**
- Add new events: **takeipbatch** and **releaseipbatch** that handle the equivalent of multiple **takeip** and **releaseip** events

takeip and **releaseip** events

Solution #1: Batch **takeip** and **releaseip**

- Add a configuration tunable so that **TAKEIP** and **RELEASEIP** *controls* get queued — so no scripts are run
- Add new controls: **TAKEIPBATCH** and **RELEASEIPBATCH**
- Add new events: **takeipbatch** and **releaseipbatch** that handle the equivalent of multiple **takeip** and **releaseip** events
- Rework the event scripts to support **takeipbatch** and **releaseipbatch**

takeip and **releaseip** events

Solution #1: Batch **takeip** and **releaseip**

- Add a configuration tunable so that **TAKEIP** and **RELEASEIP** *controls* get queued — so no scripts are run
- Add new controls: **TAKEIPBATCH** and **RELEASEIPBATCH**
- Add new events: **takeipbatch** and **releaseipbatch** that handle the equivalent of multiple **takeip** and **releaseip** events
- Rework the event scripts to support **takeipbatch** and **releaseipbatch**
- This is a lot of work...

takeip and **releaseip** events

Solution #1: Batch **takeip** and **releaseip**

- Add a configuration tunable so that **TAKEIP** and **RELEASEIP** *controls* get queued — so no scripts are run
- Add new controls: **TAKEIPBATCH** and **RELEASEIPBATCH**
- Add new events: **takeipbatch** and **releaseipbatch** that handle the equivalent of multiple **takeip** and **releaseip** events
- Rework the event scripts to support **takeipbatch** and **releaseipbatch**
- This is a lot of work...
- ...and it is not backward compatible

takeip and **releaseip** events

Solution #2: Minimise the work done in **takeip** and **releaseip**

takeip and **releaseip** events

Solution #2: Minimise the work done in **takeip** and **releaseip**

- Make running event scripts more efficient

takeip and **releaseip** events

Solution #2: Minimise the work done in **takeip** and **releaseip**

- Make running event scripts more efficient
- Allow **ipreallocated** event to do the hard work since it is only run once per node

takeip and **releaseip** events

Solution #2: Minimise the work done in **takeip** and **releaseip**

- Make running event scripts more efficient
- Allow **ipreallocated** event to do the hard work since it is only run once per node
- Force **ipreallocated** to do the hard work! :-)

takeip and **releaseip** events

Solution #2: Minimise the work done in **takeip** and **releaseip**

- Make running event scripts more efficient
- Allow **ipreallocated** event to do the hard work since it is only run once per node
- Force **ipreallocated** to do the hard work! :-)
- Find and fix bugs, annoyances, and bottlenecks

takeip and **releaseip** events

Solution #2: Minimise the work done in **takeip** and **releaseip**

- Make running event scripts more efficient
- Allow **ipreallocated** event to do the hard work since it is only run once per node
- Force **ipreallocated** to do the hard work! :-)
- Find and fix bugs, annoyances, and bottlenecks
- Can still do *solution #1* if this isn't enough... some of the hard work will already be done

Making running event scripts more efficient

vfork + exec can be cheaper than (ctdb_)fork

```
bafa467 ctdb-daemon: Deprecate RELOAD and STATUS events
7aa20cc ctdb-daemon: No need to call event scripts with CTDB_CALLED_BY_USER
2879404 ctdb-daemon: Add ctdb_vfork_with_logging()
69324b6 ctdb-daemon: Add helper process to execute event scripts
d86662a ctdb-daemon: Replace ctdb_fork_with_logging with ctdb_vfork_with_logging (part 1)
18c1f43 ctdb-daemon: Replace ctdb_fork_with_logging with ctdb_vfork_with_logging (part 2)
97575e1 ctdb-daemon: Remove unused code to run eventscripts
dd98b9d ctdb-tests: Set CTDB_EVENT_HELPER when running with local daemons
a92fd11 ctdb-daemon: Remove ctdb_fork_with_logging()

Signed-off-by: Amitay Isaacs <amitay@gmail.com>
```

Justification

- If the `ctdbd` process is large then doing `fork(2)` many times is expensive
- Instead, `vfork(2)` and `exec(3)` a small helper program
- Helper allows correct logging and termination handling

Allow **ipreallocated** event to do the hard work. . .

. . . by allowing it to know about individual IP address changes

```
885f89f ctdb-eventscripts: Allow "ipreallocated" event to know about changed IPs
b8ffb74 ctdb-eventscripts: Run winbindd ip-dropped in "ipreallocated" event
d87eb20 ctdb-eventscripts: Create Ganesha touch files in "ipreallocated" event
cee805a ctdb-eventscripts: Change policy routing to do all work in "ipreallocated"
59a08c0 ctdb-eventscripts: Make service reconfiguration depend on IP changes file
0f451e2 ctdb-eventscripts: Optimise retrieval of GPFS node number
```

Allow **ipreallocated** event to do the hard work. . .

. . . by allowing it to know about individual IP address changes

```
885f89f ctdb-eventscripts: Allow "ipreallocated" event to know about changed IPs
b8ffb74 ctdb-eventscripts: Run winbindd ip-dropped in "ipreallocated" event
d87eb20 ctdb-eventscripts: Create Ganesha touch files in "ipreallocated" event
cee805a ctdb-eventscripts: Change policy routing to do all work in "ipreallocated"
59a08c0 ctdb-eventscripts: Make service reconfiguration depend on IP changes file
0f451e2 ctdb-eventscripts: Optimise retrieval of GPFS node number
```

Analysis

1. **takeip** and **releaseip** drop information into a state file

Allow **ipreallocated** event to do the hard work. . .

. . . by allowing it to know about individual IP address changes

```
885f89f ctdb-eventscripts: Allow "ipreallocated" event to know about changed IPs
b8ffb74 ctdb-eventscripts: Run winbindd ip-dropped in "ipreallocated" event
d87eb20 ctdb-eventscripts: Create Ganesha touch files in "ipreallocated" event
cee805a ctdb-eventscripts: Change policy routing to do all work in "ipreallocated"
59a08c0 ctdb-eventscripts: Make service reconfiguration depend on IP changes file
0f451e2 ctdb-eventscripts: Optimise retrieval of GPFS node number
```

Analysis

1. **takeip** and **releaseip** drop information into a state file
2. **ipreallocated** processes this state file to make IP address specific configuration changes

Allow **ipreallocated** event to do the hard work. . .

. . . by allowing it to know about individual IP address changes

```
885f89f ctdb-eventscripts: Allow "ipreallocated" event to know about changed IPs
b8ffb74 ctdb-eventscripts: Run winbindd ip-dropped in "ipreallocated" event
d87eb20 ctdb-eventscripts: Create Ganesha touch files in "ipreallocated" event
cee805a ctdb-eventscripts: Change policy routing to do all work in "ipreallocated"
59a08c0 ctdb-eventscripts: Make service reconfiguration depend on IP changes file
0f451e2 ctdb-eventscripts: Optimise retrieval of GPFS node number
```

Analysis

1. **takeip** and **releaseip** drop information into a state file
2. **ipreallocated** processes this state file to make IP address specific configuration changes
3. Resulting simplification

Allow **ipreallocated** event to do the hard work. . .

. . . by allowing it to know about individual IP address changes

```
885f89f ctdb-eventscripts: Allow "ipreallocated" event to know about changed IPs
b8ffb74 ctdb-eventscripts: Run winbindd ip-dropped in "ipreallocated" event
d87eb20 ctdb-eventscripts: Create Ganesha touch files in "ipreallocated" event
cee805a ctdb-eventscripts: Change policy routing to do all work in "ipreallocated"
59a08c0 ctdb-eventscripts: Make service reconfiguration depend on IP changes file
0f451e2 ctdb-eventscripts: Optimise retrieval of GPFS node number
```

Analysis

1. **takeip** and **releaseip** drop information into a state file
2. **ipreallocated** processes this state file to make IP address specific configuration changes
3. Resulting simplification
4. Performance optimisation

Force **ipreallocated** event to do the hard work. . .

. . . by moving scripts that run **takeip** and **releaseip** to their own directory

```
1822c40 ctdb-daemon: IP events are considered internal events

Signed-off-by: Amitay Isaacs <amitay@gmail.com>


0fd4e0f ctdb-tests: Local daemons startup must pass --ip-event-script-dir
4b1112c ctdb-eventscripts: Split 10.interface between events.d/ and ip_events.d/

Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Force **ipreallocated** event to do the hard work...

...by moving scripts that run **takeip** and **releaseip** to their own directory

```
1822c40 ctdb-daemon: IP events are considered internal events

Signed-off-by: Amitay Isaacs <amitay@gmail.com>


0fd4e0f ctdb-tests: Local daemons startup must pass --ip-event-script-dir
4b1112c ctdb-eventscripts: Split 10.interface between events.d/ and ip_events.d/

Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Outcome

- **takeip** and **releaseip** events only run 1 script...so far...

Fix bugs, annoyances and bottlenecks

Bug #1: Who are these replies for?

```
...
2014/05/05 14:06:23.607793 [31085]: Add IP 192.168.99.3
2014/05/05 14:06:23.624186 [31085]: Add IP 192.168.99.2
2014/05/05 14:06:23.653991 [31085]: Could not find idr:493
2014/05/05 14:06:23.654032 [31085]: pnn 0 Invalid reqid 493 in ctdb_reply_control
2014/05/05 14:06:23.654045 [31085]: Could not find idr:494
2014/05/05 14:06:23.654053 [31085]: pnn 0 Invalid reqid 494 in ctdb_reply_control
...
```

Bug #1: Who are these replies for?

```
...
2014/05/05 14:06:23.607793 [31085]: Add IP 192.168.99.3
2014/05/05 14:06:23.624186 [31085]: Add IP 192.168.99.2
2014/05/05 14:06:23.653991 [31085]: Could not find idr:493
2014/05/05 14:06:23.654032 [31085]: pnn 0 Invalid reqid 493 in ctdb_reply_control
2014/05/05 14:06:23.654045 [31085]: Could not find idr:494
2014/05/05 14:06:23.654053 [31085]: pnn 0 Invalid reqid 494 in ctdb_reply_control
...
```

Analysis

- Someone reworked `ctdb reloadips` to make it send **releaseip** and **takeip** controls asynchronously...

Bug #1: Who are these replies for?

```
...
2014/05/05 14:06:23.607793 [31085]: Add IP 192.168.99.3
2014/05/05 14:06:23.624186 [31085]: Add IP 192.168.99.2
2014/05/05 14:06:23.653991 [31085]: Could not find idr:493
2014/05/05 14:06:23.654032 [31085]: pnn 0 Invalid reqid 493 in ctdb_reply_control
2014/05/05 14:06:23.654045 [31085]: Could not find idr:494
2014/05/05 14:06:23.654053 [31085]: pnn 0 Invalid reqid 494 in ctdb_reply_control
...
```

Analysis

- Someone reworked `ctdb reloadips` to make it send **releaseip** and **takeip** controls asynchronously...
- ... but they forget to register state so that the replies could be waited for!

Bug #1: Who are these replies for?

```
...
2014/05/05 14:06:23.607793 [31085]: Add IP 192.168.99.3
2014/05/05 14:06:23.624186 [31085]: Add IP 192.168.99.2
2014/05/05 14:06:23.653991 [31085]: Could not find idr:493
2014/05/05 14:06:23.654032 [31085]: pnn 0 Invalid reqid 493 in ctdb_reply_control
2014/05/05 14:06:23.654045 [31085]: Could not find idr:494
2014/05/05 14:06:23.654053 [31085]: pnn 0 Invalid reqid 494 in ctdb_reply_control
...
```

Analysis

- Someone reworked `ctdb reloadips` to make it send **releaseip** and **takeip** controls asynchronously...
- ... but they forget to register state so that the replies could be waited for!
- Who would make such a mistake?

Bug #1: Who are these replies for?

```
...
2014/05/05 14:06:23.607793 [31085]: Add IP 192.168.99.3
2014/05/05 14:06:23.624186 [31085]: Add IP 192.168.99.2
2014/05/05 14:06:23.653991 [31085]: Could not find idr:493
2014/05/05 14:06:23.654032 [31085]: pnn 0 Invalid reqid 493 in ctdb_reply_control
2014/05/05 14:06:23.654045 [31085]: Could not find idr:494
2014/05/05 14:06:23.654053 [31085]: pnn 0 Invalid reqid 494 in ctdb_reply_control
...
```

Analysis

- Someone reworked `ctdb reloadips` to make it send **releaseip** and **takeip** controls asynchronously...
- ... but they forget to register state so that the replies could be waited for!
- Who would make such a mistake?

Culprit

```
950e23f ctdbd: Make ctdb_reloadips_child send controls asynchronously
Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Fix bugs, annoyances and bottlenecks

Bug #1: Who are these replies for?

```
...
2014/05/05 14:06:23.607793 [31085]: Add IP 192.168.99.3
2014/05/05 14:06:23.624186 [31085]: Add IP 192.168.99.2
2014/05/05 14:06:23.653991 [31085]: Could not find idr:493
2014/05/05 14:06:23.654032 [31085]: pnn 0 Invalid reqid 493 in ctdb_reply_control
2014/05/05 14:06:23.654045 [31085]: Could not find idr:494
2014/05/05 14:06:23.654053 [31085]: pnn 0 Invalid reqid 494 in ctdb_reply_control
...
```

Analysis

- Someone reworked `ctdb reloadips` to make it send **releaseip** and **takeip** controls asynchronously...
- ... but they forget to register state so that the replies could be waited for!
- Who would make such a mistake?

Culprit

```
950e23f ctdbd: Make ctdb_reloadips_child send controls asynchronously
Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Fix

```
e5778cc ctdb/daemon: reloadips must register state of asynchronous controls
Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Fix bugs, annoyances and bottlenecks

Bug #2: Why is **releaseip** still running after **deleteip** finishes?

```
...
2014/05/05 14:07:20.744574 [31085]: Delete IP 192.168.99.199
2014/05/05 14:07:20.749204 [31085]: Delete IP 192.168.99.200
2014/05/05 14:07:20.848403 [recoverd:31275]: Reenabling takeover runs
2014/05/05 14:07:20.856375 [31085]: 10.interface: Kept secondary 192.168.99.197/24 on dev lo
2014/05/05 14:07:20.857044 [31085]: 10.interface: Kept secondary 192.168.99.191/24 on dev lo
...
```

Bug #2: Why is **releaseip** still running after **deleteip** finishes?

```
...
2014/05/05 14:07:20.744574 [31085]: Delete IP 192.168.99.199
2014/05/05 14:07:20.749204 [31085]: Delete IP 192.168.99.200
2014/05/05 14:07:20.848403 [recoverd:31275]: Reenabling takeover runs
2014/05/05 14:07:20.856375 [31085]: 10.interface: Kept secondary 192.168.99.197/24 on dev lo
2014/05/05 14:07:20.857044 [31085]: 10.interface: Kept secondary 192.168.99.191/24 on dev lo
...
```

Analysis

- It has always been like this.

Bug #2: Why is **releaseip** still running after **deleteip** finishes?

```
...
2014/05/05 14:07:20.744574 [31085]: Delete IP 192.168.99.199
2014/05/05 14:07:20.749204 [31085]: Delete IP 192.168.99.200
2014/05/05 14:07:20.848403 [recoverd:31275]: Reenabling takeover runs
2014/05/05 14:07:20.856375 [31085]: 10.interface: Kept secondary 192.168.99.197/24 on dev lo
2014/05/05 14:07:20.857044 [31085]: 10.interface: Kept secondary 192.168.99.191/24 on dev lo
...
```

Analysis

- It has always been like this.
- Each **deleteip** control invokes a **releaseip** event *asynchronously*...

Bug #2: Why is **releaseip** still running after **deleteip** finishes?

```
...
2014/05/05 14:07:20.744574 [31085]: Delete IP 192.168.99.199
2014/05/05 14:07:20.749204 [31085]: Delete IP 192.168.99.200
2014/05/05 14:07:20.848403 [recoverd:31275]: Reenabling takeover runs
2014/05/05 14:07:20.856375 [31085]: 10.interface: Kept secondary 192.168.99.197/24 on dev lo
2014/05/05 14:07:20.857044 [31085]: 10.interface: Kept secondary 192.168.99.191/24 on dev lo
...
```

Analysis

- It has always been like this.
- Each **deleteip** control invokes a **releaseip** event *asynchronously*...
- ...and *does not wait!*

Bug #2: Why is **releaseip** still running after **deleteip** finishes?

```
...
2014/05/05 14:07:20.744574 [31085]: Delete IP 192.168.99.199
2014/05/05 14:07:20.749204 [31085]: Delete IP 192.168.99.200
2014/05/05 14:07:20.848403 [recoverd:31275]: Reenabling takeover runs
2014/05/05 14:07:20.856375 [31085]: 10.interface: Kept secondary 192.168.99.197/24 on dev lo
2014/05/05 14:07:20.857044 [31085]: 10.interface: Kept secondary 192.168.99.191/24 on dev lo
...
```

Analysis

- It has always been like this.
- Each **deleteip** control invokes a **releaseip** event *asynchronously*...
- ...and *does not wait!*
- That's a little bit unexpected...

Bug #2: Why is **releaseip** still running after **deleteip** finishes?

Fix

```
commit 9b907536fb657fa15c02858caf0ffff633ecd478
Author: Martin Schwenke <martin@meltin.net>
Date:   Wed Jan 22 13:30:47 2014 +1100

    ctdb/daemon: Make delete IP wait until the IP is released

    reloadips really expects deleted IPs to be released before completing.
    Otherwise the recovery daemon starts failing the local IP check.  The
    races that follow can cause a node to be banned.

    To make the error handling simple, do the actual deletion in
    release_ip_callback().
```

Fix bugs, annoyances and bottlenecks

Bug #2: Why is **releaseip** still running after **deleteip** finishes?

Fix

```
commit 9b907536fb657fa15c02858caf0ffff633ecd478
Author: Martin Schwenke <martin@meltin.net>
Date:   Wed Jan 22 13:30:47 2014 +1100

    ctdb/daemon: Make delete IP wait until the IP is released

    reloadips really expects deleted IPs to be released before completing.
    Otherwise the recovery daemon starts failing the local IP check.  The
    races that follow can cause a node to be banned.

    To make the error handling simple, do the actual deletion in
    release_ip_callback().
```

Optimisation

```
20c7196 ctdb/daemon: Optimise deletion of IPs
Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Fix bugs, annoyances and bottlenecks

Bug #2: Why is **releaseip** still running after **deleteip** finishes?

Fix

```
commit 9b907536fb657fa15c02858caf0ffff633ecd478
Author: Martin Schwenke <martin@meltin.net>
Date:   Wed Jan 22 13:30:47 2014 +1100

    ctdb/daemon: Make delete IP wait until the IP is released

    reloadips really expects deleted IPs to be released before completing.
    Otherwise the recovery daemon starts failing the local IP check.  The
    races that follow can cause a node to be banned.

    To make the error handling simple, do the actual deletion in
    release_ip_callback().
```

Optimisation

```
20c7196 ctdb/daemon: Optimise deletion of IPs
Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Tweak

```
6cdde27 ctdb:daemon avoid goto ctdb_remove_orphaned_ifaces()
Signed-off-by: Gregor Beck <gbeck@sernet.de>
```

Fix bugs, annoyances and bottlenecks

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

Fix bugs, annoyances and bottlenecks

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs

Fix bugs, annoyances and bottlenecks

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each
4. Wait for outstanding controls

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each
4. Wait for outstanding controls
5. Enable IP allocation runs

Fix bugs, annoyances and bottlenecks

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each
4. Wait for outstanding controls
5. Enable IP allocation runs
6. Trigger an IP allocation run

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each
4. Wait for outstanding controls
5. Enable IP allocation runs
6. Trigger an IP allocation run

- All actual IP address manipulation should be done in the IP allocation run

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each
4. Wait for outstanding controls
5. Enable IP allocation runs
6. Trigger an IP allocation run

- All actual IP address manipulation should be done in the IP allocation run
- However, **deleteip** now waits for IP addresses to be released, so work is done there

Fix bugs, annoyances and bottlenecks

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each
4. Wait for outstanding controls
5. Enable IP allocation runs
6. Trigger an IP allocation run

- All actual IP address manipulation should be done in the IP allocation run
- However, **deleteip** now waits for IP addresses to be released, so work is done there

Fix

```
d9defb9 ctdb-daemon: Deletion of IPs is deferred until the next takeover run
Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each
4. Wait for outstanding controls
5. Enable IP allocation runs
6. Trigger an IP allocation run

- All actual IP address manipulation should be done in the IP allocation run
- However, **deleteip** now waits for IP addresses to be released, so work is done there

Fix

```
d9defb9 ctdb-daemon: Deletion of IPs is deferred until the next takeover run
Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Analysis

- Not upstream yet

Fix bugs, annoyances and bottlenecks

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each
4. Wait for outstanding controls
5. Enable IP allocation runs
6. Trigger an IP allocation run

- All actual IP address manipulation should be done in the IP allocation run
- However, **deleteip** now waits for IP addresses to be released, so work is done there

Fix

```
d9defb9 ctdb-daemon: Deletion of IPs is deferred until the next takeover run
Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Analysis

- Not upstream yet
- Changes behaviour of `ctdb delip`...

Fix bugs, annoyances and bottlenecks

Annoyance #1: **deleteip** doesn't fit the `ctdb reloadips` model

Analysis

`ctdb reloadips` does:

1. Disable IP allocation runs
2. Determine public IP addresses that are *no longer configured* and do asynchronous **deleteip** control for each
3. Determine public IP addresses that are *newly configured* and do asynchronous **addip** control for each
4. Wait for outstanding controls
5. Enable IP allocation runs
6. Trigger an IP allocation run

- All actual IP address manipulation should be done in the IP allocation run
- However, **deleteip** now waits for IP addresses to be released, so work is done there

Fix

```
d9defb9 ctdb-daemon: Deletion of IPs is deferred until the next takeover run
Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Analysis

- Not upstream yet
- Changes behaviour of `ctdb delip`... but `ctdb reloadips` is recommended

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great
- However, when a primary IP address is deleted Linux deletes all of the secondaries... and they need to be re-added

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great
- However, when a primary IP address is deleted Linux deletes all of the secondaries... and they need to be re-added
- In the worst case that's $O(n!)$

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great
- However, when a primary IP address is deleted Linux deletes all of the secondaries... and they need to be re-added
- In the worst case that's $O(n!)$

Solutions

- Use /32 netmasks to avoid secondaries?

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great
- However, when a primary IP address is deleted Linux deletes all of the secondaries... and they need to be re-added
- In the worst case that's $O(n!)$

Solutions

- Use /32 netmasks to avoid secondaries? No, that breaks broadcast...

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great
- However, when a primary IP address is deleted Linux deletes all of the secondaries... and they need to be re-added
- In the worst case that's $O(n!)$

Solutions

- Use /32 netmasks to avoid secondaries? No, that breaks broadcast...
- Batch IP address releases — when primary is being deleted then just delete it (first) else do something clever to minimise re-adds...

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great
- However, when a primary IP address is deleted Linux deletes all of the secondaries... and they need to be re-added
- In the worst case that's $O(n!)$

Solutions

- Use /32 netmasks to avoid secondaries? No, that breaks broadcast...
- Batch IP address releases — when primary is being deleted then just delete it (first) else do something clever to minimise re-adds...
- `# echo 1 > /proc/sys/net/ipv4/conf/all/promote_secondaries`

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great
- However, when a primary IP address is deleted Linux deletes all of the secondaries... and they need to be re-added
- In the worst case that's $O(n!)$

Solutions

- Use /32 netmasks to avoid secondaries? No, that breaks broadcast...
- Batch IP address releases — when primary is being deleted then just delete it (first) else do something clever to minimise re-adds...
- `# echo 1 > /proc/sys/net/ipv4/conf/all/promote_secondaries`
- Thanks to:
  `http://tmartiro.blogspot.com.au/2013/03/remove-primary-address-without-removing.html`

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great
- However, when a primary IP address is deleted Linux deletes all of the secondaries... and they need to be re-added
- In the worst case that's $O(n!)$

Solutions

- Use /32 netmasks to avoid secondaries? No, that breaks broadcast...
- Batch IP address releases — when primary is being deleted then just delete it (first) else do something clever to minimise re-adds...
- `# echo 1 > /proc/sys/net/ipv4/conf/all/promote_secondaries`
- Thanks to:
  `http://tmartiro.blogspot.com.au/2013/03/remove-primary-address-without-removing.html`
- Hmmm... this is undocumented...

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Analysis

- The above improvements are great
- However, when a primary IP address is deleted Linux deletes all of the secondaries... and they need to be re-added
- In the worst case that's $O(n!)$

Solutions

- Use /32 netmasks to avoid secondaries? No, that breaks broadcast...
- Batch IP address releases — when primary is being deleted then just delete it (first) else do something clever to minimise re-adds...
- `# echo 1 > /proc/sys/net/ipv4/conf/all/promote_secondaries`
- Thanks to:
  `http://tmartiro.blogspot.com.au/2013/03/remove-primary-address-without-removing.html`
- Hmmm... this is undocumented...
- How long has it been there?

Bottleneck #1: Re-adding secondary address...

promote_secondaries introduced in Linux kernel ...

```
commit 8f937c6099858eee15fae14009dcbd05177fa91d
Author: Harald Welte <laforge@gnumonks.org>
Date:   Sun May 29 20:23:46 2005 -0700

    [IPV4]: Primary and secondary addresses

    Add an option to make secondary IP addresses get promoted
    when primary IP addresses are removed from the device.
    It defaults to off to preserve existing behavior.

$ git describe 8f937c6099858eee15fae14009dcbd05177fa91d
v2.6.12-rc5-193-g8f937c6
```

Bottleneck #1: Re-adding secondary address...

promote_secondaries introduced in Linux kernel ...

```
commit 8f937c6099858eee15fae14009dcbd05177fa91d
Author: Harald Welte <laforge@gnumonks.org>
Date:   Sun May 29 20:23:46 2005 -0700

    [IPV4]: Primary and secondary addresses

    Add an option to make secondary IP addresses get promoted
    when primary IP addresses are removed from the device.
    It defaults to off to preserve existing behavior.

$ git describe 8f937c6099858eee15fae14009dcbd05177fa91d
v2.6.12-rc5-193-g8f937c6
```

promote_secondaries documented in Linux kernel...

```
commit d922e1cb1ea17ac7f0a5c3c2be98d4bd80d055b8
Author: Martin Schwenke <martin@meltin.net>
Date:   Tue Jan 28 15:26:42 2014 +1100

    net: Document promote_secondaries

$ git describe d922e1cb1ea17ac7f0a5c3c2be98d4bd80d055b8
v3.13-8616-gd922e1c
```

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Use promote_secondaries in CTDB

```
176ae6c ctdb-eventscripts: Deleting IPs should use the promote_secondaries option
058e14c ctdb-eventscripts: Fix regression in IP add/delete functions
20eb250 ctdb-eventscripts: Pass event name to service_reconfigure()
aceb341 ctdb-eventscripts: Minimise the work done by policy routing

Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Use promote_secondaries in CTDB

```
176ae6c ctdb-eventscripts: Deleting IPs should use the promote_secondaries option
058e14c ctdb-eventscripts: Fix regression in IP add/delete functions
20eb250 ctdb-eventscripts: Pass event name to service_reconfigure()
aceb341 ctdb-eventscripts: Minimise the work done by policy routing

Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Analysis

- Upstream in samba/master, not in ctdb/2.5 yet

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Use promote_secondaries in CTDB

```
176ae6c ctdb-eventscripts: Deleting IPs should use the promote_secondaries option
058e14c ctdb-eventscripts: Fix regression in IP add/delete functions
20eb250 ctdb-eventscripts: Pass event name to service_reconfigure()
aceb341 ctdb-eventscripts: Minimise the work done by policy routing

Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Analysis

- Upstream in samba/master, not in ctdb/2.5 yet
- Oops!

Fix bugs, annoyances and bottlenecks

Bottleneck #1: Re-adding secondary address...

Use promote_secondaries in CTDB

```
176ae6c ctdb-eventscripts: Deleting IPs should use the promote_secondaries option
058e14c ctdb-eventscripts: Fix regression in IP add/delete functions
20eb250 ctdb-eventscripts: Pass event name to service_reconfigure()
aceb341 ctdb-eventscripts: Minimise the work done by policy routing

Signed-off-by: Martin Schwenke <martin@meltin.net>
```

Analysis

- Upstream in samba/master, not in ctdb/2.5 yet
- Oops!
- Policy routing should never lose unintended routes...

Questions?

Legal Statement

- This work represents the view of the author and does not necessarily represent the view of IBM.
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others.