dns_proxy.md

DNS Proxy

In a typical hub/spoke deployment, its very common for the WAN interfaces to have some sort of dynamic interface such as DHCP, PPPoE, LTE etc. The 128T router can dynamically learn the DNS server address for such interfaces and can load balance the DNS requests across those servers. The dns-proxy feature aims to provide a simple way to proxy all DNS requests originating on the LAN side to the learned server address(es) on the WAN side without having to re-configure or update the client endpoints. This will allow the network to better adapt to failures on the WAN interfaces while minimizing loss of connectivity from client side applications.

Overview

The common workflow for using this feature is as follows:

- Advertise/Configure a DNS server for the LAN network(s)
- Configure a DNS proxy service to match the advertisement
- Configure the service-route to indicate the WAN interface(s) to be used for proxying the DNS requests

This document discusses the various options for the workflow above to provide general guidance on how to use this feature.

Advertise Interface Address as DNS Server

A key component for DNS proxy feature is the ability to configure a fixed address as the DNS address for the clients on the LAN side. A typical choice is to use the 128T interface address as the DNS server address, though the feature is not limited to this choice. The selected address can either be statically configured on the clients or configured via DHCP server (either external or 128T acting as the DHCP server). On a linux based test system this was done via updating the `/etc/resolv.conf` file as shown below.

```
# cat /etc/resolv.conf
; generated by /usr/sbin/dhclient-script
search openstacklocal
nameserver 10.10.10.1
#
```

Configuring DNS proxy service

The special `dns-proxy` application-type is used for creating a DNS proxy service. All the usual service configuration such as access-policy etc is applicable to this service. The `dns-proxy` application-type indicates to the 128T router to perform a destination NAT on the traffic when the session is created for the service.

```
admin@node1.conductor# show config running authority service lan-dns-proxy

config

    authority

        service  lan-dns-proxy
            name              lan-dns-proxy

            transport         udp
                protocol    udp

                port-range  53
                    start-port  53
                exit
            exit
            address           10.10.10.1

            access-policy     lan
                source      lan
                permission  allow
            exit
```

```
            access-policy     _internal_
                source      _internal_
                permission  allow
            exit
            application-type  dns-proxy
        exit
    exit
exit

admin@node1.conductor#
```

The example configuration captures all DNS traffic sent to address `10.10.10.1` interface as configured on the test client in the previous step.

How to proxy DNS requests originating from the linux host of the 128T router

The `_internal_` tenant has a special meaning on the 128T routers as it represents the traffic originating from the host OS of the router. When the service allows the `_internal_` tenant and a `service-route` is created for this service, the target router linux environment is automatically configured for use with the DNS proxy. The `/etc/resolv.conf` is modified to point to a loopback address within the 128T router.

```
# cat /etc/resolv.conf
; This file has been automatically updated by 128T - DO NOT EDIT
nameserver 169.254.127.126
```

This allows all DNS queries (for example, as a result of dnf lookups etc) to be intercepted by 128T router and create sessions appropriately.

Configuring Service Route(s)

When the `service-route > next-hop` for the dns-proxy service points to a dynamic interface such as DHCP based interface, any learned DNS address(es) will be automatically used as destination nat target for sessions for that service. This is accomplished by populating the `next-hop > target-address` configuration internally upon address resolution. An example of the service-route configuration is as follows:

```
admin@node1.conductor# show config running authority router router1 node node1 device-interface inband-mgmt

config

    authority

        router  router1
            name  router1

            node  node1
                name              node1

                device-interface  inband-mgmt
                    name               inband-mgmt
                    pci-address        0000:00:03.0

                    network-interface  inband-mgmt-intf
                        name          inband-mgmt-intf
                        global-id     1
                        conductor     true
                        tenant        wan
                        source-nat    true

                        host-service  ssh
                            service-type  ssh
                        exit

                        host-service  web
                            service-type  web
                        exit

                        dhcp          v4
```

```
                    exit
                exit
            exit
        exit
    exit
exit

admin@node1.conductor#
admin@node1.conductor# show config running authority router router1 service-route dns-proxy-route

config

    authority

        router  router1
            name           router1

            service-route  dns-proxy-route
                name          dns-proxy-route
                service-name  lan-dns-proxy

                next-hop      node1 inband-mgmt-intf
                    node-name  node1
                    interface  inband-mgmt-intf
                    target-address 8.8.8.8
                exit
            exit
        exit
    exit
exit

admin@node1.conductor#
```

A few key points about the service-route for a dns-proxy service type:

Multiple learned DNS addresses

If the dynamic interface learns multiple IP addresses, the 128T router will apply a round-robin load-balancing strategy amongst those IP address. Here's how you can check the details on the learned DNS addresses.

```
admin@node1.conductor# show dhcp v4 router router1 name inband-mgmt-intf detail
Sat 2020-02-22 04:43:12 UTC

============================================================
 Router
============================================================
  Node:                             node1
    Device Interface:               inband-mgmt
      Network Interface:            inband-mgmt-intf
        Dhcp State:                 Resolved
          State Machine State:      Bound
          Lease Start Time:         Sat Feb 22 00:28:30 2020
          Lease Renewal Time:       Sat Feb 22 12:28:30 2020
          Lease Rebinding Time:     Sat Feb 22 18:28:30 2020
          Lease Expiration Time:    Sun Feb 23 00:28:30 2020
          Learned MTU:              0 bytes
          Server Address:           192.168.1.2
          Dns Server Address:
            - 172.20.0.100
            - 172.20.0.101
          Addresses:
            Address:                192.168.1.10
            Prefix Length:          24
            Gateway:                192.168.1.1

Completed in 0.14 seconds
admin@node1.conductor#
```

The following example illustrates how the round-robin strategy gets applied for load-balancing the data across multiple learned addresses for two back-to-back queries.

```
admin@node1.conductor# show sessions router router1 | grep dns-proxy
✔ Piping output...
 745809ea-7ada-4225-a379-4159c1e226a2   fwd   lan-dns-proxy                                    lan          dpdk2
0   udp     10.10.10.11          52513   10.10.10.1              53   192.168.1.10         16398   false
4   0 days  0:00:03
 745809ea-7ada-4225-a379-4159c1e226a2   rev   lan-dns-proxy                                    lan
inband-mgmt      0   udp     172.20.0.100            53   192.168.1.10          16398   0.0.0.0                  0
false                     4   0 days  0:00:03
admin@node1.conductor#
admin@node1.conductor#
admin@node1.conductor# show sessions router router1 | grep dns-proxy
✔ Piping output...
 801d9495-66a4-44cf-9eea-e22731389a95   fwd   lan-dns-proxy                                    lan          dpdk2
0   udp     10.10.10.11          44426   10.10.10.1              53   192.168.1.10         16399   false
5   0 days  0:00:03
 801d9495-66a4-44cf-9eea-e22731389a95   rev   lan-dns-proxy                                    lan
inband-mgmt      0   udp     172.20.0.101            53   192.168.1.10          16399   0.0.0.0                  0
false                     5   0 days  0:00:03
admin@node1.conductor#
```

Manually configured `target-address`

As seen in the example above the `service-route > next-hop` points to a DHCP interface and also specifies `8.8.8.8` as target-address. When such manual configuration is present, all the learned address are combined with the statically configured address(es). Based on the previous example, it means that we now have three valid DNS server targes viz. `8.8.8.8, 172.20.0.100, 172.20.0.101`. This configuration also allow the user to configure a failsafe DNS server address in case the DHCP server did not provide any valid DNS server addresses.

Conclusion

The dns-proxy feature can be used to greatly simply the network topoplogy while dyanmically reacting to chagnes or failures to the upstream WAN interfaces.