ECO connectivity failover and improved policy control

Background and problem statement

Currently the network traffic from the EVE microservices (zedagent, logmanager, downloader, etc) can operate with active/active handling of multiple management ports. However, traffic from an ECO (using a network instance) can not reliably get such connectivity except when using the mesh overlay network instances. A local network instance can be specified to use the tag "uplink" which results in all of the management ports being used, but due to the constraints of NAT this attempt at active/active connectivity does not work reliably.

Furthermore, when there are multiple management ports with different preferences (for instance a free Ethernet plus a pay-per-use LTE connection) the level of policy control extends to the EVE microservices. This, subject to policy control, should also apply to ECO traffic.

Proposal

Active/standby for network instances

Instead of using active/active for network instances, it is a lot simpler to use active/standby. This means that a given local (or cloud) network instance is using a single port at any given time, but as the EVE microservices detects failures the network instance will failover to using a different uplink.

The NIM (network interface manager) microservice is already testing the connectivity towards the controller, and is also detecting when complete local connectivity is lost (the cable being unplugged and/or the DHCP lease expiring). That checking and testing results in updates to the `DeviceNetworkStatus` object. That testing mechanism will be extended to test all of the management ports, and report the lastSucceeded/lastFailed timestamps in the `DeviceNetworkStatus` per each port.

The zedrouter microservice (which implements the network instances) will observe these changes and use them to update the network instance's use of the port, and the associated NAT rules (MASQUERADE and port map rules) to be associated with the currently used management port. Also, the PBR rules for the source IP address of the network instance (shown in ip rule show) will be updated to refer to the table associated with the currently used management port.

This means that should a network instance switch from using eth0 to wwan0 all of the outbound traffic will start flowing via wwan0, and the portmap for inbound connectivity will only apply to wwan0. As such, a user doing a ssh into the Edge Container will need to reconnect to the IP address on the wwan0 interface since the eth0 has presumable failed.

Note that as part of these changes we can remove the maintenance of 500 routing table, which has been used for all NAT traffic.

Priority handling

When zedrouter notices that the currently used management port for a network instance has failed, then it needs to select an alternate. That should be performed the same way as the logic in zedcloud/send.go, which is to first look at the management ports marked as "free" and only if none of this are found to be working (i.e., lastSucceeded timestamp is more than lastFailed), then proceed to look at the management ports with free=false.

Refined priority handling

Currently the notion of a free management interface is specified in the API in PhyIOUsagePolicy, which is directly attached to the provisioning of the device ports. Longer term that might not be sufficient along two axes:

- The boolean free/non-free is not able to capture multiple levels of failover from e.g., Ethernet, to LTE, and to satellite connectivity as a last resort.
- There might be a need to update the policy dynamically once a device has been deployed. For instance, a device on a ship might default to using LTE for baseimage and edge container image downloads, but when it leaves port there might be a need to allow image downloads over satellite to deploy new edge containers.

Hence introducing a notion of a priority/preference which can be set/changed for each management ports makes sense, with zero corresponding to the current "free" and increasing numbers are different levels of backup connectivity. In addition, one should be able to specify which operations can be performed over which priority to that the user can e.g., say

- baseimage download OK for priority 0 and 1
- edge container image download OK for priority 0
- for each network instance specify the priorities, e.g., the default network instance OK for priority 0, 1 and 2.

These policy and priority handling enhancements will be explored in more detail.

Implementation notes (order of deliverables):

1) NIM should produce NITS (Network Instance Test Status) based on IP being available on ports. NIM should also respond to any changes to ports/IP addresses (UP/DOWN/Address change) and pick one from the ports available to network instance. In this phase, NIM only checks if there is an IP address assigned to port and would deem it usable if it does have one assigned. NITS status should have network instance and the port name that should be used for sending out traffic.

Zedrouter should then respond to NITS and re-configure/program NAT rules, port map rules etc corresponding to applications attached to the network instance.

2) Enhance NITS test to use Zedcloud ping to find any remote network failures.

3) Assign priorities to ports and produce NITS results based on priorities. Free port flag that is present currently will be replaced with priority relative to other ports. NITS should give the highest priority working port for a given network instance.

4) Network instances should allow user to configure a test target URL/IP address along with test timer interval. EVE device instead of using the default Zedcloud ping, will use the configured test URL/IP to test if a given port can be used.