Public Networks in Quantum

Link: ***https://blueprints.launchpad.net/quantum/+spec/quantum-v2-public-networks***
Target Milestone: Folsom-3

Rationale

At the moment, all Quantum networks are assumed to be private, meaning that the tenant which creates the network owns it, and is the only one authorized to create ports for interfaces on that network.
Unfortunately this model does not adapt very well to some common scenarios in cloud infrastructure, where instances are often attached to one or more 'public' networks, which are owned by the service providers and used by the majority of tenants.
The aim of this blueprint is to introduce the 'public network' concept in Quantum, intended as networks which are owned by a single tenant, but where all the tenants are allowed to create ports for their virtual interfaces.

Design details

A public network in Quantum does not differ from any other network in terms of data model. In order to distinguish public networks from private networks a specific attribute will be used (`public`, boolean). This attribute will be part of the core Quantum API.

When a port is created on a public network, the tenant will be allowed to specify all the attributes which are normally specified, with the exception of the following attributes:

- fixed_ips: as the subnet is shared among all tenants, an user does not have a mean to understand which IPs are available. Also, a malicious user might use this attribute to know the IP addresses of other tenants' VMs currently deployed on the same subnet.
- mac_address: even if the chance of clashes are probably limited, the same kind of concerns we had for fixed_ips applies also to mac_address.
- host_routes: according to the way they will be implemented for Quantum v2 API, we will decide whether it would make sense to allow tenants to specify host_routes. However, disallowing users to control these attributes on public networks make probably sense.

NOTE: The behaviour of the API w.r.t. the above listed attributes has not yet been decided. They might either be ignored, or the API might explicitly return a 403 error. In the latter case, the quantum policy framework should allow for specifying rules for disallowing non-owner tenants to specify the above listed attributes on POST and PUT requests for ports on public networks.

For the Folsom release we are proposing an "all-or-nothing" model, with a limited and fixed delegation mechanism.

- By 'all-or-nothing' we mean that a network is either private or completely public, i.e.:

shared with all the tenants;

- By limited delegation mechanism, we mean that the owner of the network also owns all dependent objects, except the ports used by other tenants' VMs[1].
    - Quantum API will allow tenants to operate on port they own without checking for ownership of networks and subnets, assuming the network is public[2].
    - Quantum API will allow any tenant to list public networks.
    - Quantum API will also allow tenants to list ports for public networks, but only ports owned by the tenant issuing the request will be returned.
    - Quantum API will also enable read operation on subnets for a public network.

- By fixed delegation mechanism, we mean that Quantum API will not provide any mechanism for extending delegations, eg: giving the right to create and manipulate subnets to tenants different from the owner.

API Changes

The only important API change probably concerns the way in which networks are retrieved from the Quantum v2 API.
Considering that there's a good analogy between networks and images, the way in which image services handle public and private images has been analyzed. Primary focus was on Glance, but also S3 (AWS & Eucalyptus) and Cloudstack were taken into account.[3]

It seems a filter attribute for GET /networks would be the best solution in terms of simplicity for the final user and ease of implementation. Currently we are considering several possible alternatives:

1. Single flag: `public { True | False (default) }`: If `public=True`, only public networks will be returned; `tenant_id` will be ignored. Specifying `public=False` without a `tenant_id` filter will always return no results.
   PRO: Directly Maps to underlying model
   CONS: Two API calls required for retrieving all the networks for a given tenant
2. Two flags: `public { True | False (default) }`, `private { True (default) | False }`. The private flag allows for returning all the networks a tenant

[1] An alternative, simpler, design with no delegation mechanism was considered. The issue with this model is that there would be no way for discriminating ports belonging to different tenants on a public network. As a result, it would have been quite difficult preventing tenants from deleting other tenants' ports or avoiding other tenants' ports to be listed in GET operations on /ports.
[2] A possible alternative design is whether the subnet is tenant-owned just like the port. On the plus side, the model will probably be slightly more consistent; however on the minus side it is probably not a good idea to give tenants freedom of managing the IP space on a public network. For instance a tenant might create a subnet 16.0.0.0/4, thus taking for herself 1/16 of the whole address space!
[3] Cloudstack networking API has also been considered. The list_networks operation allows for filtering on network categories, such as 'system', through querystring filters.

has access to, both private and public, in a single API call.
PRO: Single API call
CONS: Does not directly match data model, filters would need to be OR-ed when both flags are specified, where by default they are AND-ed.

3. Single enum flag: `network_type { public | private | both (default) }`. Functionally equivalent to alternative #2, but uses a single flag.|
PRO: Single API call
CONS: Does not directly match data model

Preference at the moment would be for option #1, unless a simple solution for implementing option #3 in Quantum v2 API is devised.

The table below depicts the API view for a tenant operating on a public network.

/v2.0/networks			
Attribute	Type	Required	CRUD
id	uuid-str	N/A	R
name	str	N/A	R
admin_state_up	bool	N/A	R
status	str	N/A	R
tenant_id	uuid-str	N/A	R
subnets	list(uuid)	N/A	R
public	bool	N/A	R
/v2.0/subnets			
id	uuid-str	N/A	R
network_id	uuid-str	N/A	R
ip_version	int(4,6)	N/A	R
cidr	str	N/A	R
gateway_ip	str	N/A	R
dns_nameservers	list(str)	N/A	R
allocation_pools	list(dict)	N/A	R
additional_host_routes	list(dict)	N/A	?
/v2.0/ports			
id	uuid-str	Y	R
network_id	uuid-str	Y	CR
admin_state_up	bool	N	CRUD
status	str	N	R
mac_address	str	N	R
fixed_ips	list(dict)	N	R
host_routes	list(dict)	N	?

device_id	str	N	CRUD

Access control

We will not have extended access control for public networks in the Folsom release.
The owner-only model which is currently implemented will continue to apply, with the only exception that if a network is marked as “public”, then each tenant will be allowed to create ports on it.

A public network is available to all tenants. However in principle the creation of a public network should be an ‘administrative’ operation, meaning that the ‘public’ attribute should be read-only for regular tenants. In this way, only tenants with the admin role will be able to create public networks.

Nova integration

The API class for interfacing nova with Quantum will need some changes to behave differently when a public network is specified, as it is currently assumed that all the networks where VIFs for instance will be plugged belong to the tenant. We will need to ensure that public networks are supported too.

Changes are specifically required in the `allocate_for_instance` method of the class `nova.network.quantumv2.api`. As of the current implementation, this method requires that all the networks requested for any new instance are owned by the same tenant for which the instance is created. This has to be related in order to ensure quantum ports can be created on public networks too.

This can be achieved by filing a bug and having it merged after implementing the feature on the Quantum side; this kind of change will probably not require a blueprint.