Document revisions history

Version	Date	Amendment
1.0	12/15/23	
1.1	1/4/2023	- GW client will get heartbeats from monitor, and must commit suicide if missing few. - Added TOC

Background

The nvmeof GW should support high availability. High availability means that even in the case that a certain GW is down, there will be another available path for the initiator to be able to continue the IO through another GW. It means that initially there are at least 2 paths in which the nvme initiator can use to do the IO to the namespace(s). The multi pathing is achieved by connecting to the subsystem through more than 1 GW. This is native to the nvme initiator behavior, and this is done by connecting the nvme initiator to all relevant GWs (e.g. nvme connect-all command). Multi pathing allows the option to the initiator, to use one of the paths to write to the subsystem. This is a must for HA, but not enough. The problem is that the initiator should not simultaneously write to the same namespace(s) (i.e. volumes). Writing simultaneously to the same namespace(s) will eventually result in data inconsistency because there is no guarantee on the order of the writes that arrive at the namespace via the different GWs. There are many design options to solve this issue, the selected option that we implemented, is discussed here.
The core idea is to provide an Active-Standby access from the initiator to namespace(s). It means that at any point in time, there is only one (and only one) active path from the initiator to a namespace, but there are also standby path(s). The management of the Active-Standby states is being done in a new component that is called **NVMeofGwMon**.
Namespaces in nvme belong to a subsystem. That's why the management of the entire Active-Standby states is done at a subsystem level. The implementation is using the nvme ANA protocol, which allows to define a state for each path. The state can be Optimized, Inaccessible, or Non-optimized. In our implementation, we set the state to either Optimized (i.e. Active), or Inaccessible (i.e. Standby). The ANA protocol is using ANA groups to define the path states. So per path, we can see different ANA groups, and per ANA group, we can know if the path is Optimized or Inaccessible. ANA group is a collection of namespaces.

The **NVMeofGwMon** should manage the ANA groups in a way that a particular group is alway optimized on at **only** one path (i.e. GW), and it is Inaccessible on **all** the other paths (i.e. GWs). The **NVMeofGwMon** needs to track the liveliness of all the GWs, and handle cases like:

1. GW disappeared.
2. GW reappeared.

The **NVMeofGwMon** should take the required actions when such events occur. E.g.

1. GW disappeared - the **NVMeofGwMon** should assign a new GW to be Optimized on this path, and then it needs to update all the GWs in the group, to change their state accordingly. This is called Failover.
2. GW reappeared - the **NVMeofGwMon** should re-assign the returning GW to be Optimized on this path, and then it needs to update all the GWs in the group, to change their state accordingly. This is called Failback.

Main design decisions

HA environment setup requirements

It is assumed that between the nvmeof initiator (i.e. the nvmeof client) and the nvmeof target (i.e. the nvmeof ceph gw), there is full redundancy in the network connectivity. This means that the nvmeof initiator has 2 ethernet ports that are connected to the nvmeof target, via a network with redundancy (e.g. 2 networks switches).

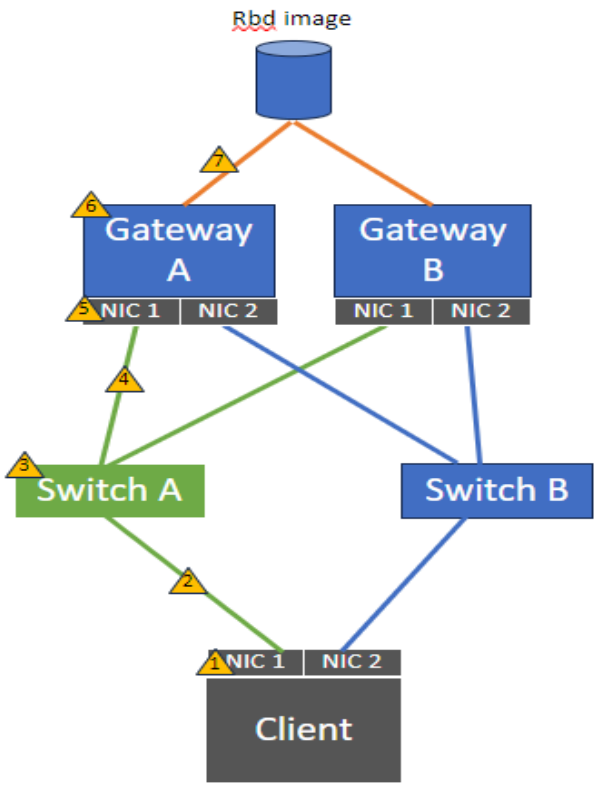


Figure 1 - Full redundancy in the network connectivity

Failover scenarios

The HA mode is not taking care of situations where the network paths between the nvmeof initiator and the nvmeof target are broken. This case should be covered by the network configuration which includes full redundancy to the network paths.

The following failover scenarios will be taken care of by the HA mode:

1. GW dead.
2. GW removed by cephadm.
3. GW has no active listener (i.e. all listeners removed).
4. Network partition between the gateway and rbd

Blocklisting

Whenever we failover a path, there is a danger that the peer that owned this path before, might still be alive, or might be temporarily frozen, and it might still hold some inflight IOs that it is about to submit to Ceph. This might cause data

inconsistencies, and therefore we will always blocklist the peer before taking over any path. Blocklist will invalidate any inflight IO that it has.

Subsystems

The HA mode is determined when creating a new NVMe subsystem using the gRPC or the CLI. If the HA mode is set to True, then the ANA states on this subsystem will be managed by the **NVMeofGwMon.** It is not allowed to manipulate the ANA states externally (e.g. via the SPDK RPC), because doing that will invalidate the auto HA solution.

ANA groups

The HA solution will only use ANA group 1..number of active GWs. It means that if we have 2 GWs, we will use ANA groups 1 and 2, and if we have 3 GWs we will use ANA groups 1,2,3, and so on. The idea is that each GW will always own one ANA grp, and will be standby on the other ANA groups.

Load Balancing

The optimal load balancing will be achieved when the number of active (i.e. optimized) namespaces, is distributed evenly between all of the GWs. It means that every GW will handle the same number of namespaces in a good path IO situation (where all GWs are up and running). To achieve that, the gRPC and CLI asks to define the load balancing group when creating a namespace. The load balancing group is translated into an ANA grp id. It is the responsibility of the user, to divide the namespaces evenly between the load balancing groups. This assignment is persistent in the OMAP state, and can be modified by another gRPC/CLI call.

GW initialization

The nvmeof GW initialization is changing. The GW must get some initial data from the **NVMeofGwMon** to be able to complete its initialization. The initial data will include the ANA grp id that it should own. Based on the ANA grp id, the GW can tell which unique controller ids to use, and it knows on which ANA grp id it should be optimized. This means that the GW initialization sequence is delayed until it gets this initial data. And until this initial data is received, the gRPC/CLI and the SPDK initialization is on hold.

Network partition

It is possible that the nvmeof GW monitor will think that a GW is down, but in reality the GW will be alive. This can happen in a case of a network partition for example. The problem in this case is that the monitor will decide to failover the ANA groups that “belong” to this GW, to other GWs. But the GW will not know about it. For this reason, it is decided that the GW (i.e. the GW client in this case), will get heartbeats from the monitor every few seconds. In the case that the heartbeats stop (i.e. not heartbeat few cycles), the GW will commit suicide to avoid the case that the same ANA group is considered to be optimized by more than one GW.

Last listener removed from a GW on a subsystem

If there is a GW

Failover trigger and handling

Failback trigger and handling

Modules

There are changes in the Ceph code, and there are changes in the nvmeof GW code.

Ceph code - new modules

MNVMeofGwMap
Description: Class that coordinates Gateway's Failover/ Failback
Main responsibilities:
Coordinates the behavior of all Gateways in the CEPH that configured in HA mode.
Implements stateful behavior for performing Failover/Failback by Gateways within the same subsystem.
Supports for independent state machines(per ANA group) within the same Gateway.
Implements the blocklist of ceph entries used for blocked traffic related to specific ANA groups.
Holds GWMAP map, GW_Created map database.

MNVMeofGwBeacon
Description:
Main responsibilities:

NVMeofGwMon
Description: New monitor in the Paxos environment - used for monitoring Gateways in HA mode
Main responsibilities:
Forwards inherited Paxos messages, aggregates the NVMeofGwMap object
Distributes maps to Paxos and broadcasts them to the GW Clients
Handles Beacon messages from the GW Clients, determines the Keep Alive timeout from the GW
Conveys CEPH commands - create/delete GW to the NVMeofGwMap embedded object
Sends immediate unicast Beacon Ack message as response to Beacon to ensure symmetric handshake

NVMeofGw
Description:
Main responsibilities:

NVMeofGwClient
Description:
Main responsibilities:

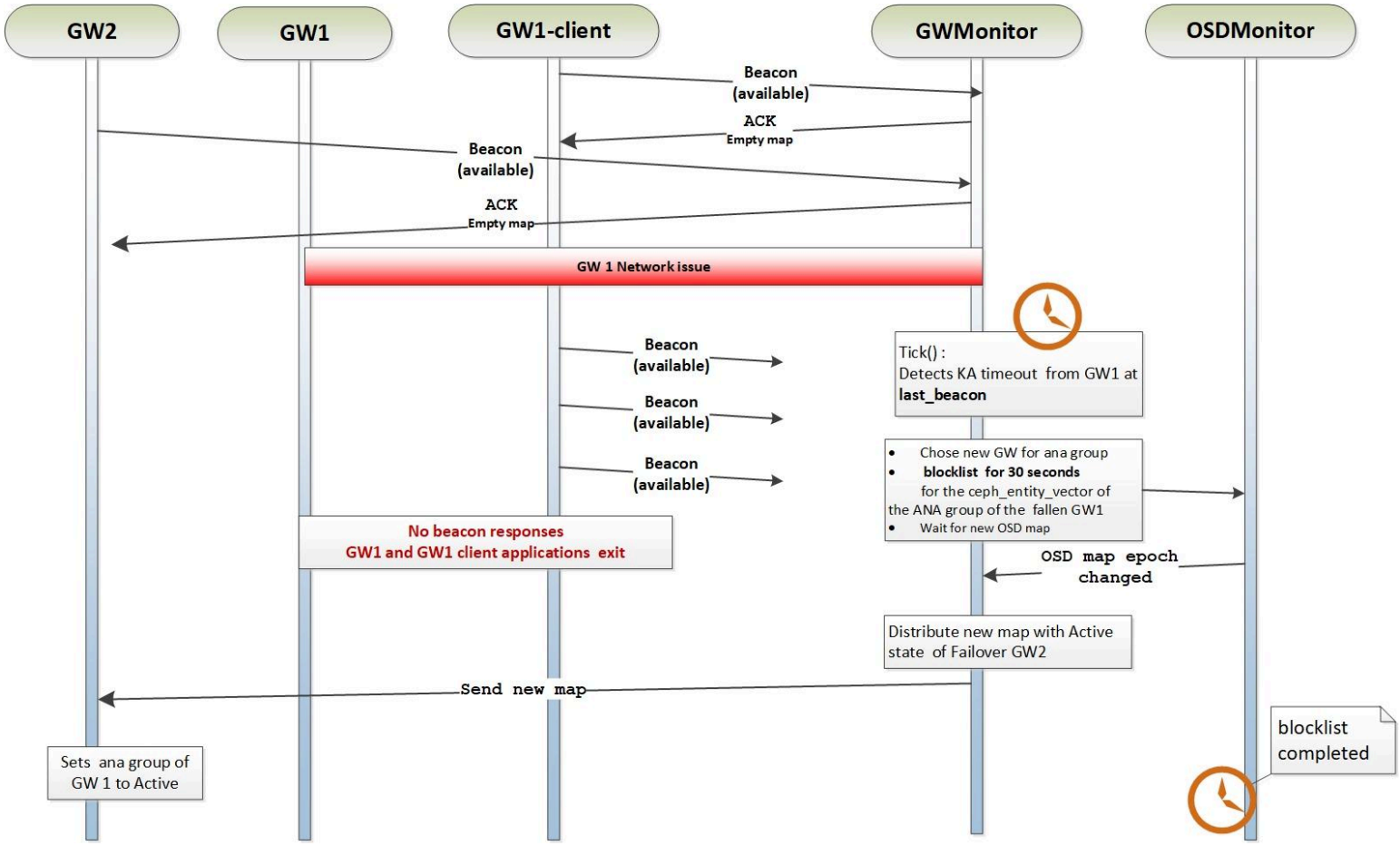
NVMeofGwMonitorGroupClient
Description:
Main responsibilities:

Ceph code - changed modules

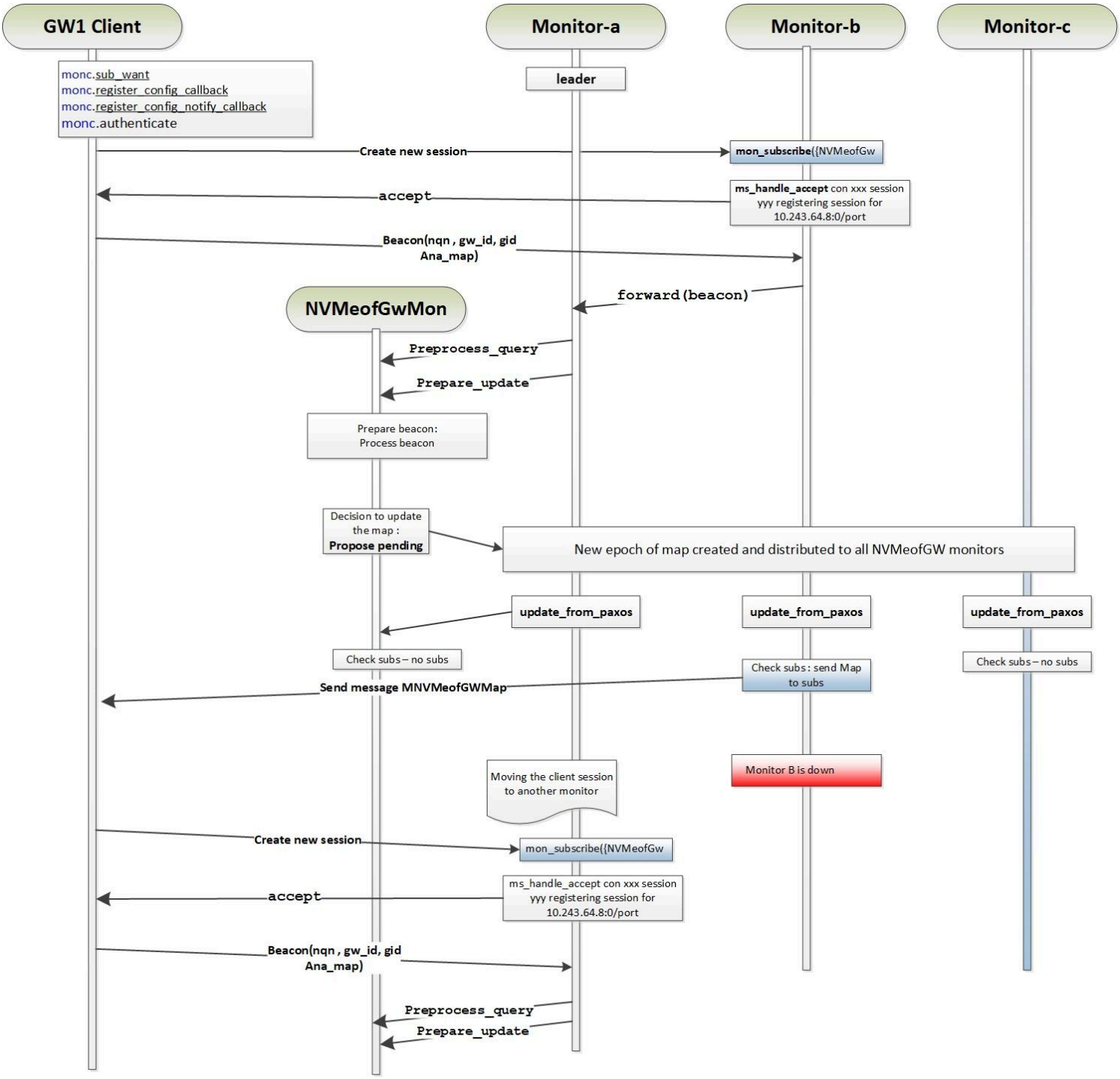
MonCommands
Monitor
Message
Nvmeof GW code - changed modules

Sequence diagrams

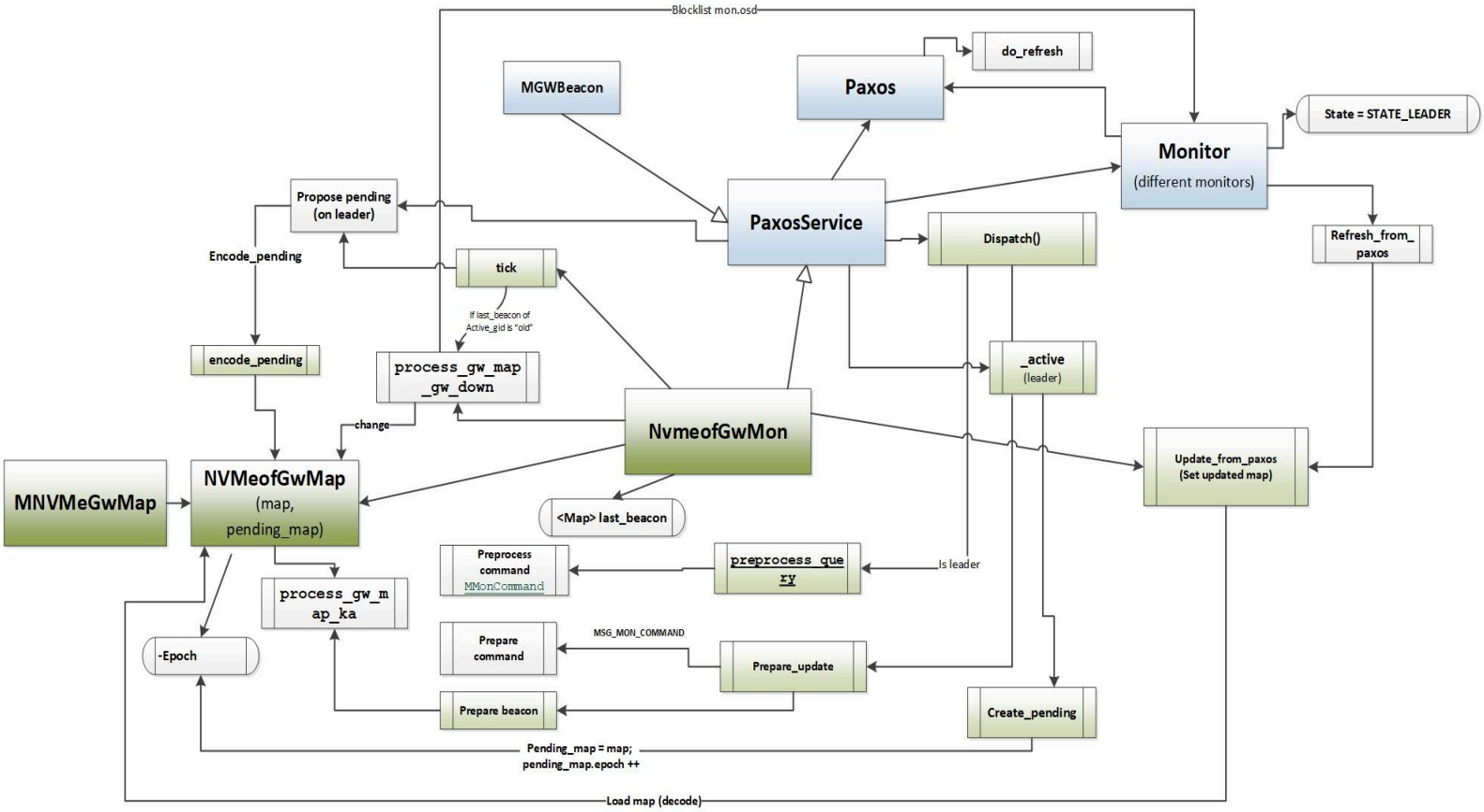
1. ceph adm deployment.
2. gw startup.
3. failover.
4. failback.
5. gw is removed from deployment.
6. network partition.



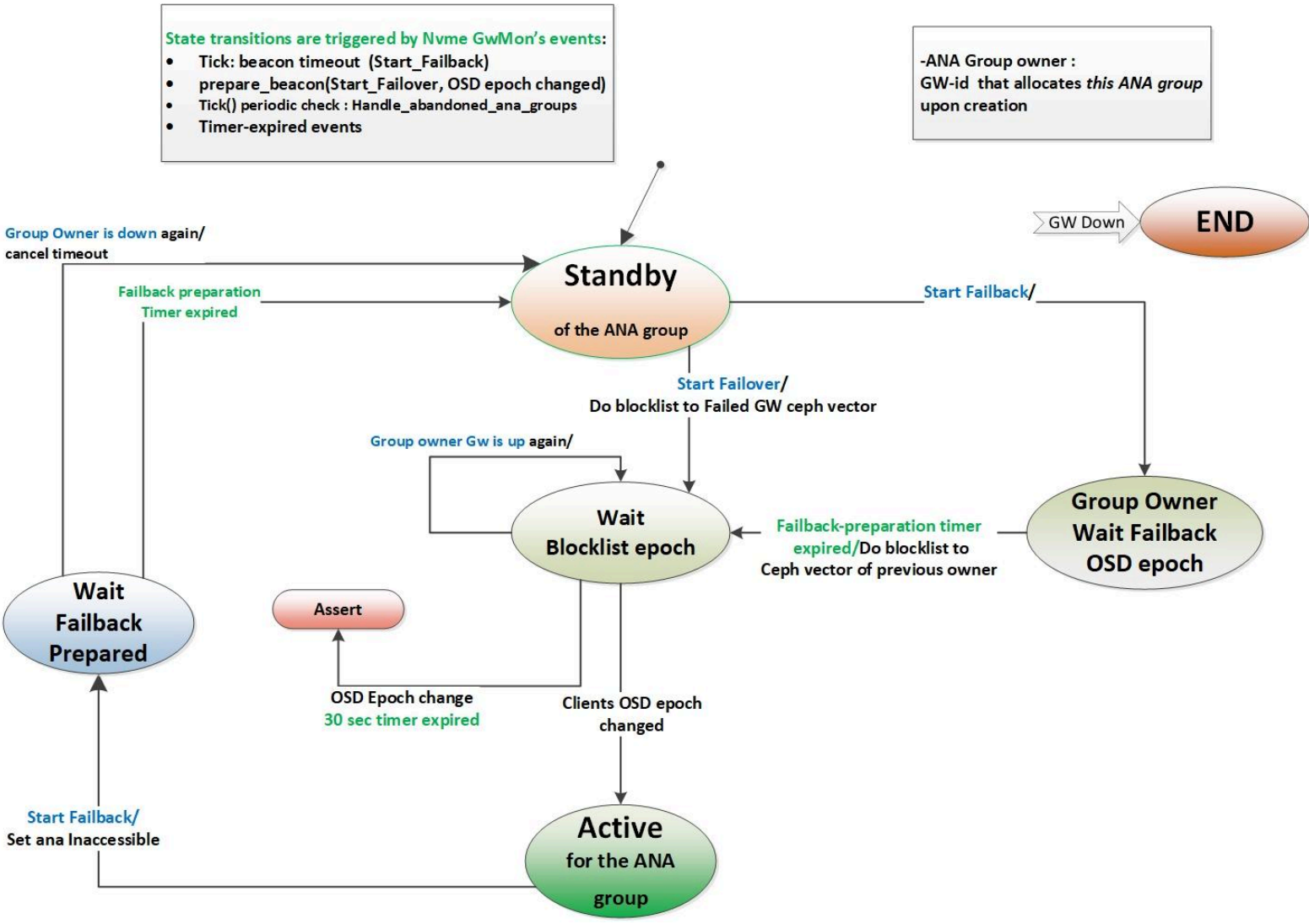
7. Last listener removed from a subsystem
8. Concurrent blocklists to the same fallen gateway
9. Gateway-Client session creation , Beacon routing, Map distribution to clients


GW1 Client
Monitor-a
Monitor-b
Monitor-c
monc.sub_want
monc.register_config_callback
monc.register_config_notify_callback
monc.authenticate
leader
Create new session
mon_subscribe({NVMeofGw
accept
ms_handle_accept con xxx session yyy registering session for 10.243.64.8:0/port
Beacon(nqn , gw_id, gid Ana_map)
NVMeofGwMon
forward (beacon)
Preprocess_query
Prepare_update
Prepare beacon: Process beacon
Decision to update the map : Propose pending
New epoch of map created and distributed to all NVMeofGW monitors
update_from_paxos
update_from_paxos
update_from_paxos
Check subs – no subs
Check subs : send Map to subs
Check subs – no subs
Send message MNVMeofGWMap
Monitor B is down
Moving the client session to another monitor
Create new session
mon_subscribe({NVMeofGw
accept
ms_handle_accept con xxx session yyy registering session for 10.243.64.8:0/port
Beacon(nqn , gw_id, gid Ana_map)
Preprocess_query
Prepare_update

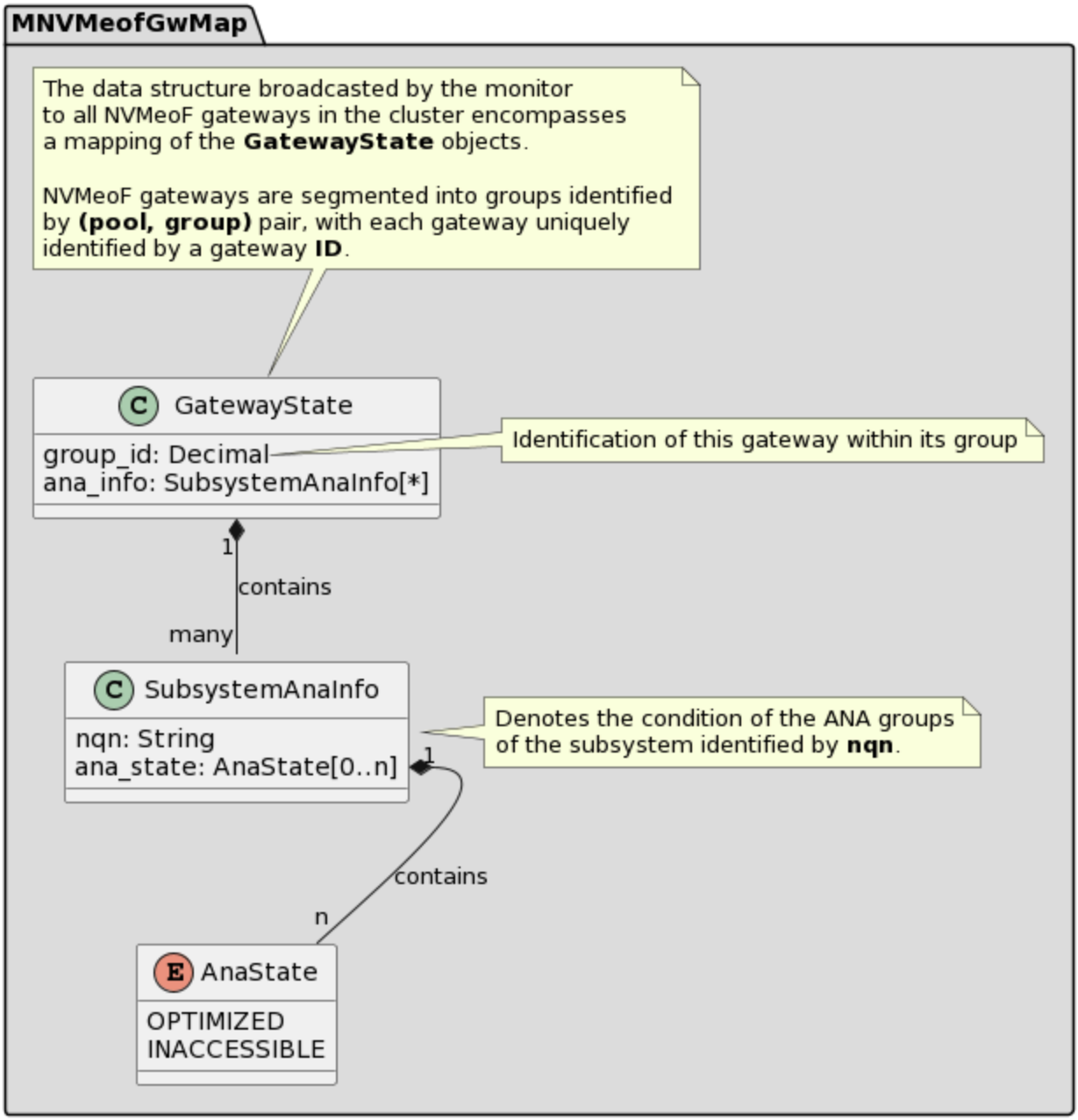
Class diagrams

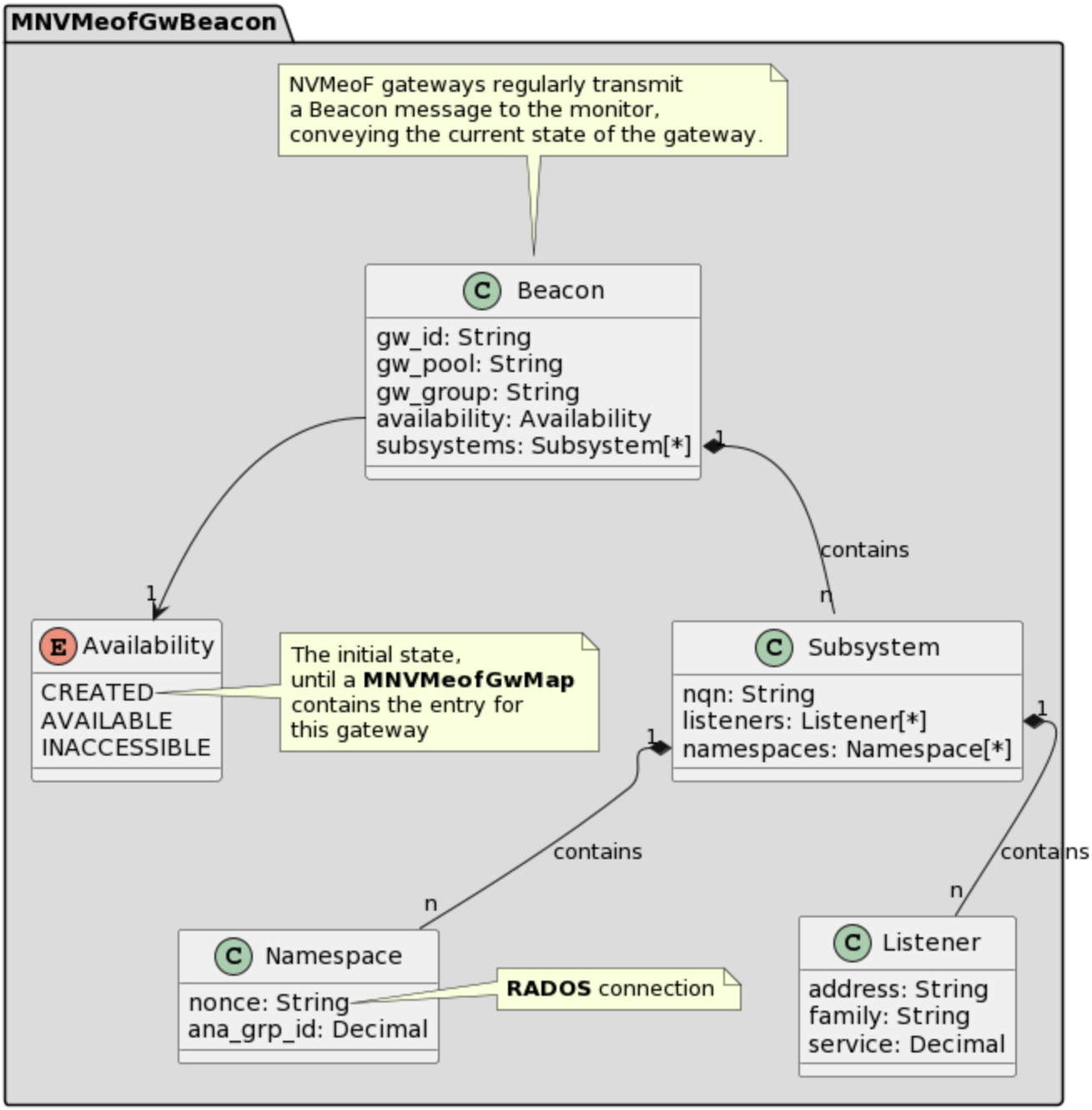


State diagrams



Messages


MNVMeofGwMap
The data structure broadcasted by the monitor
to all NVMeoF gateways in the cluster encompasses
a mapping of the **GatewayState** objects.
NVMeoF gateways are segmented into groups identified
by **(pool, group)** pair, with each gateway uniquely
identified by a gateway **ID**.
C GatewayState
group_id: Decimal
ana_info: SubsystemAnaInfo[*]
Identification of this gateway within its group
1
contains
many
C SubsystemAnaInfo
nqn: String
ana_state: AnaState[0..n]
Denotes the condition of the ANA groups
of the subsystem identified by **nqn**.
1
contains
n
E AnaState
OPTIMIZED
INACCESSIBLE


MNVMeofGwBeacon
NVMeoF gateways regularly transmit
a Beacon message to the monitor,
conveying the current state of the gateway.
Beacon
gw_id: String
gw_pool: String
gw_group: String
availability: Availability
subsystems: Subsystem[*]
1
contains
n
1
Availability
CREATED
AVAILABLE
INACCESSIBLE
The initial state,
until a MNVMeofGwMap
contains the entry for
this gateway
Subsystem
nqn: String
listeners: Listener[*]
namespaces: Namespace[*]
1
1
contains
n
contains
n
Namespace
nonce: String
ana_grp_id: Decimal
RADOS connection
Listener
address: String
family: String
service: Decimal

Map data structures

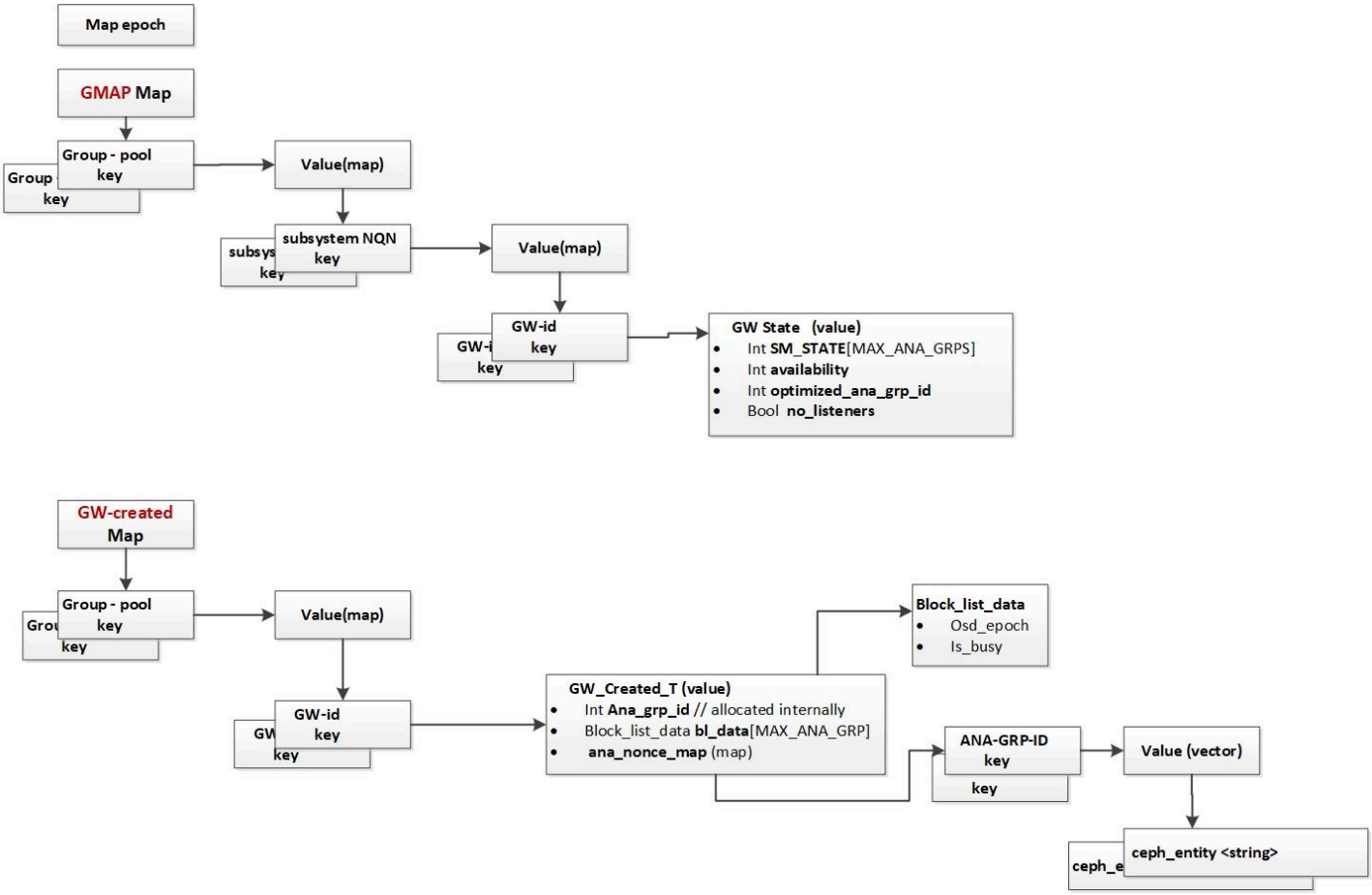
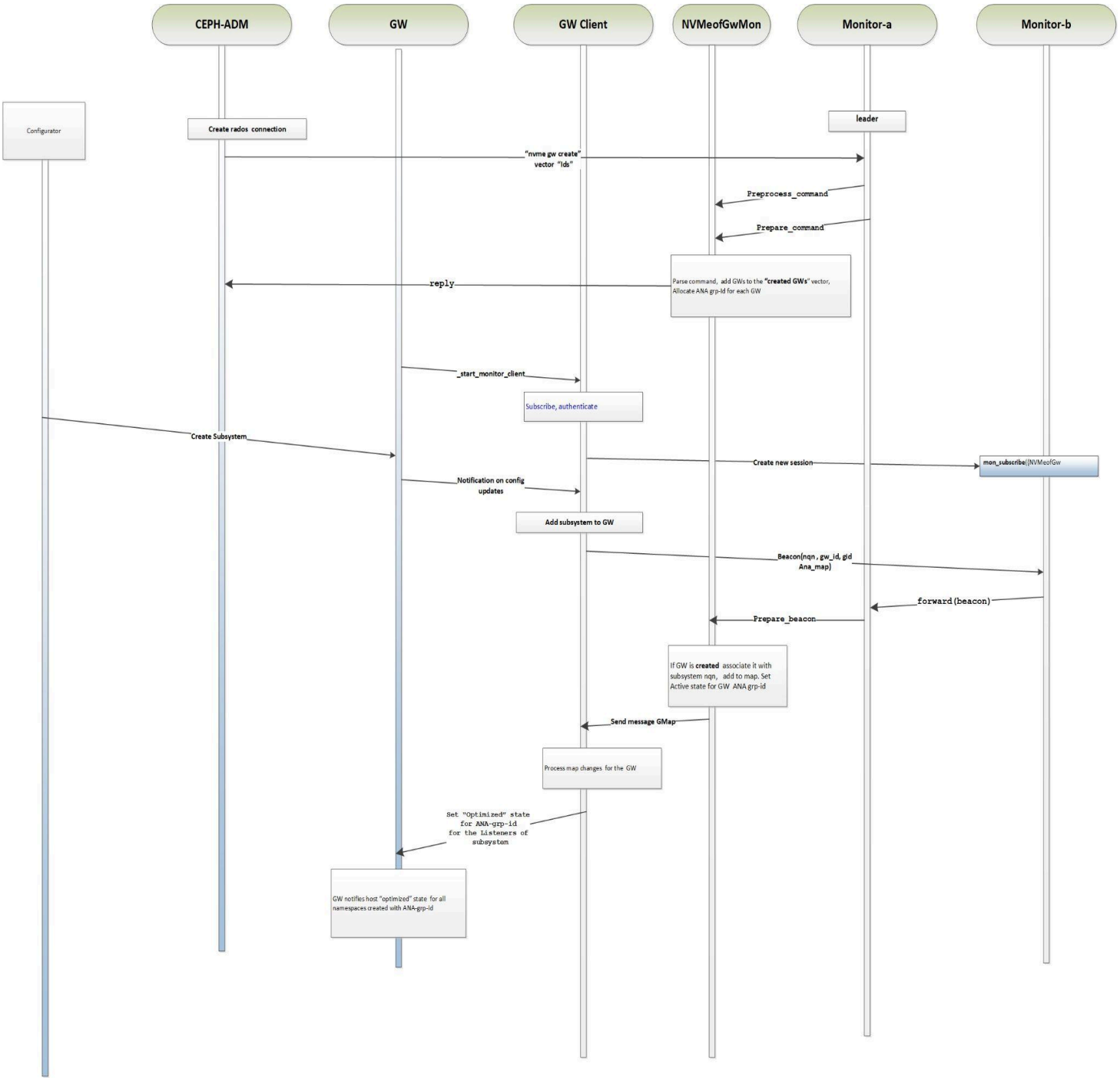
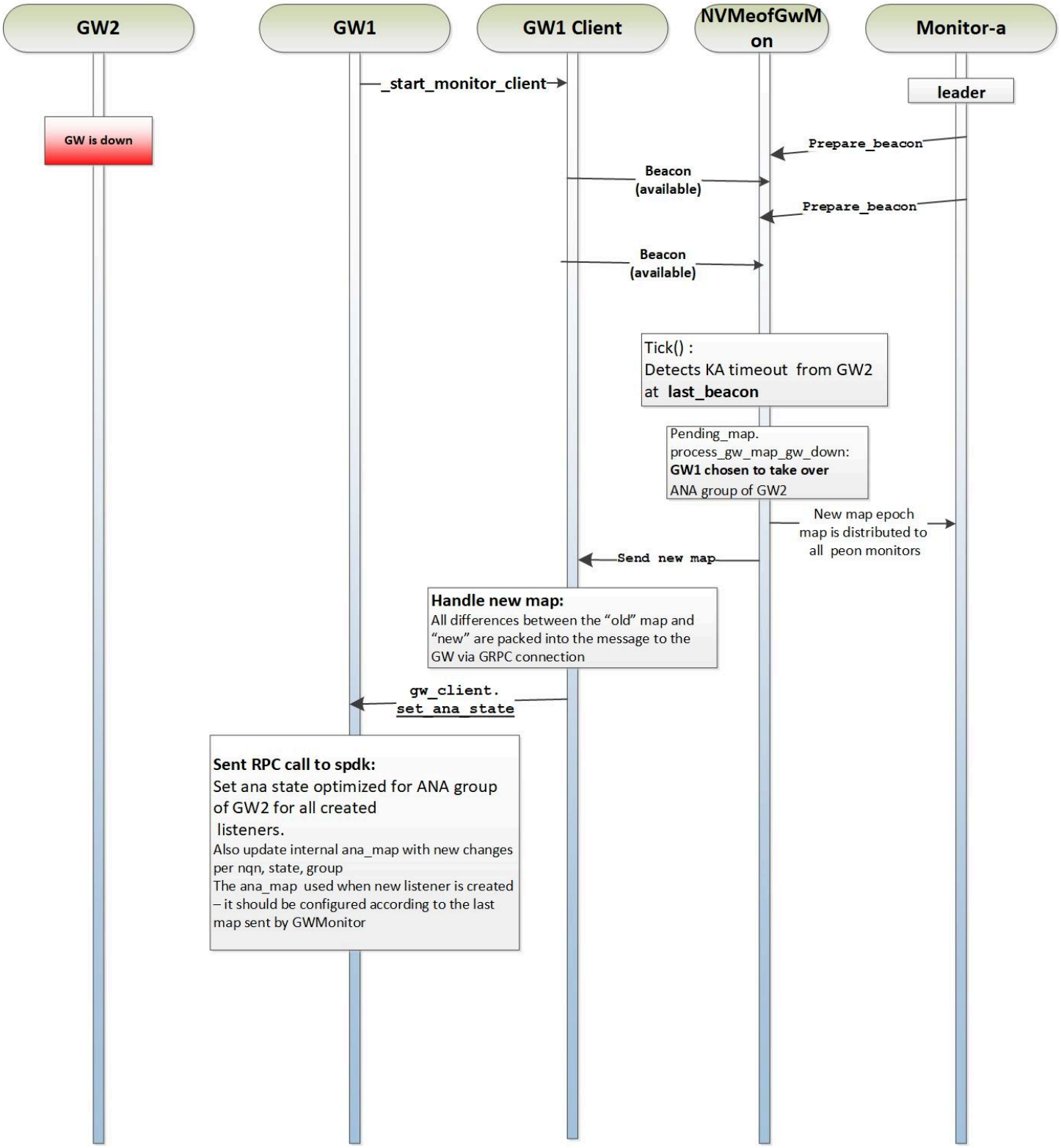


Diagram : CEPH ADM creates GW in monitor, GW client sends first beacon and receives updated map



The next diagram is Failover started after GW1 is down


GW2
GW1
GW1 Client
NVMeofGwMon
Monitor-a
_start_monitor_client
leader
GW is down
Prepare_beacon
Beacon
(available)
Prepare_beacon
Beacon
(available)
Tick() :
Detects KA timeout from GW2
at last_beacon
Pending_map.
process_gw_map_gw_down:
GW1 chosen to take over
ANA group of GW2
New map epoch
map is distributed to
all peon monitors
Send new map
Handle new map:
All differences between the "old" map and
"new" are packed into the message to the
GW via GRPC connection
gw_client.
set_ana_state
Sent RPC call to spdk:
Set ana state optimized for ANA group
of GW2 for all created
listeners.
Also update internal ana_map with new changes
per nqn, state, group
The ana_map used when new listener is created
– it should be configured according to the last
map sent by GWMonitor

Initial Integration of Ceph cluster

1. Bootstrap + add cluster node (result - 2 nodes Ceph cluster). - waiting for build.
    a. Starts nvmeofmon on the 2 nodes.
    b. Take the build from https://quay.ceph.io/repository/ceph-ci/ceph?tab=tags and Filter Tags "ceph-nvmeof-mon" (need to be in VPN, and push the image to a public repository)

2. Cephadm Deploy nvmeof GW on 2 nodes.
    a. change controller ids range.

3. Management command - nvme-gw create (gw1-id, gw2-id)
    a. Leonid to build stand alone app to send the command. - need to build stand alone application.
    b. We need to get GWs IDs for the command (consult with Gil).
    c. We might need to verify that we have monitor permissions to the rados ids - Alex.
4. Nvmeof CLI - create subsystem (with enable HA), add listeners to both GWs, and create 2 names spaces (one in ANA grp1, one in ANA grp2).
5. Take down GW1, and see that we get the RPC call to GW2 (set ana info).
    a. Still remaining items to complete the set ana info

We need to get 2 VMs to bring up this configuration. - Aviv and Barak

Integration with vstart cluster

Todo and open issues

- [x] ~~Prepare a "draft PR be uploaded to github.com/ceph/ceph so Patrick can have an initial review.~~ https://github.com/ceph/ceph/pull/54671
- [ ] Prepare the updated sequence diagrams
- [x] ~~Complete the Notify to GW API (set ANA state)~~
- [ ] Need to change SPDK to get the nonce.
- [x] ~~Need to think about Failback issue with blocklist (cluster context per ANA grp?)~~
- [x] ~~Need to think about multiple GW groups, maybe a different monitor per group is the simple solution?~~
- [ ] Summarize the main design decisions that we made so far.
- [ ] Add class diagrams if we have something available
- [ ] Add state diagrams if we have something available
- [ ] Check that we know to send commands to monitor

This diagram depicts how GW client opens session, session retain, message routing - beacons and map distribution