Voltha High Availability Design Document

Proxy & Core

Khenaidoo Nursimulu, Sergio Slobodrian

April 10, 2018

Contents

Contents

Fundamental Design Principles

Fundamental Design Principles

- **Ensure 5 or 6 9's availability**
  - 5 9's – Less than 5 min / system / year outage
  - 6 9's – Less than 30 sec / system / year outage
- **Support horizontal scaling**
- **Support 50ms recovery from failure in most cases**
- **Never lose a transaction**

Fundamental principles → Design decisions

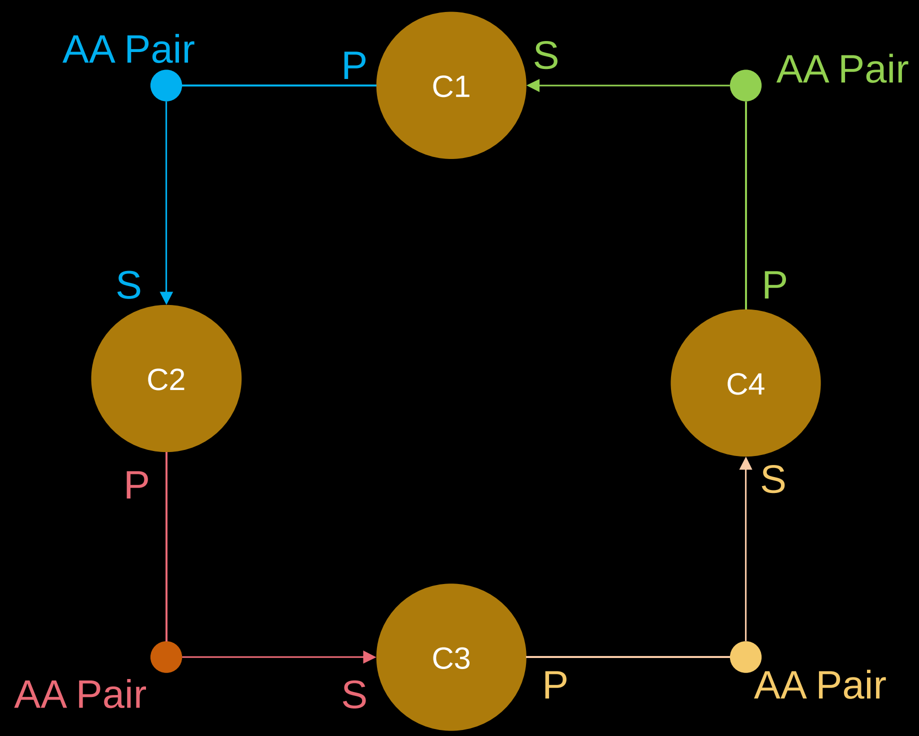
Principle	Design Decisions
Ensure 5 or 6 9's availability	Primary/Secondary Cores
Support horizontal scaling	Stateless cores
Support 50ms recovery from failure in most cases	Primary/Secondary cores, stateless cores
Never lose a transaction	Primary/Secondary cores

ciena

Proxy Design

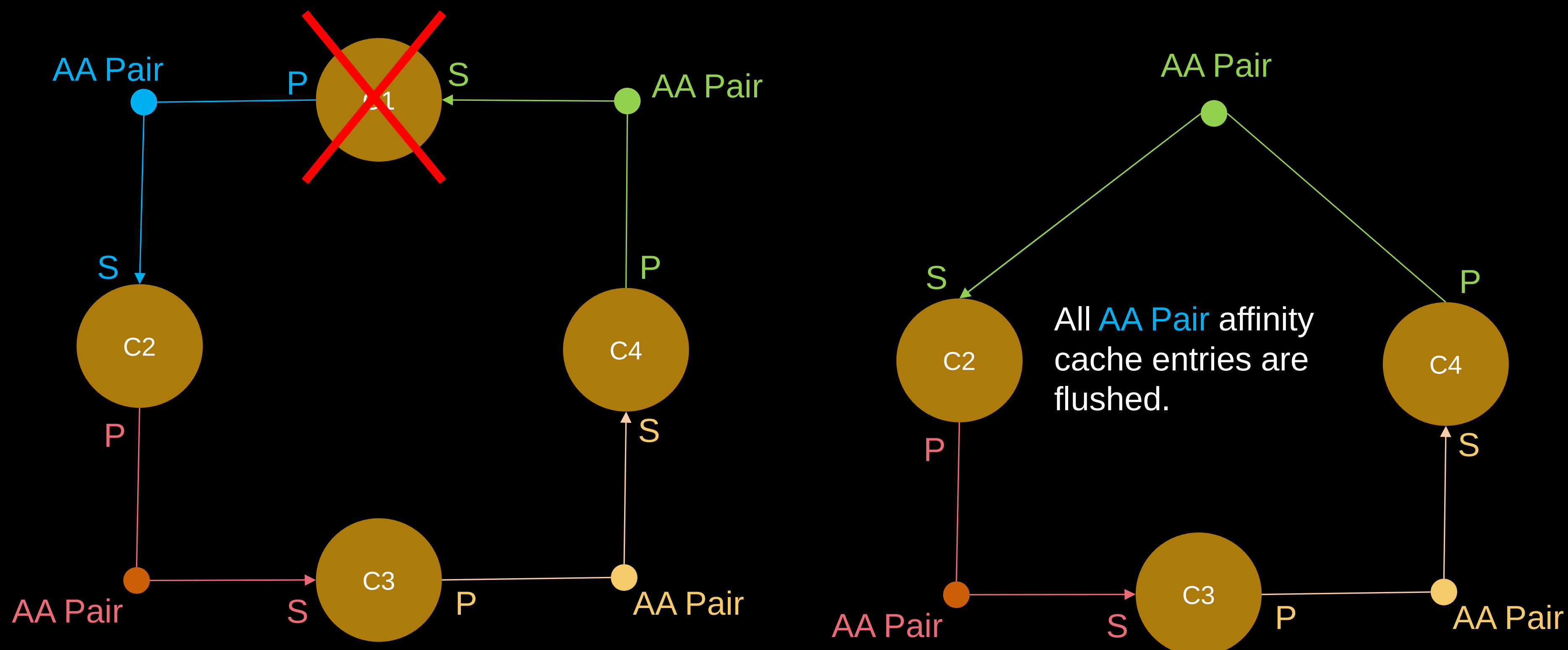
Core Addition and Removal

Active-Active Core Pairing

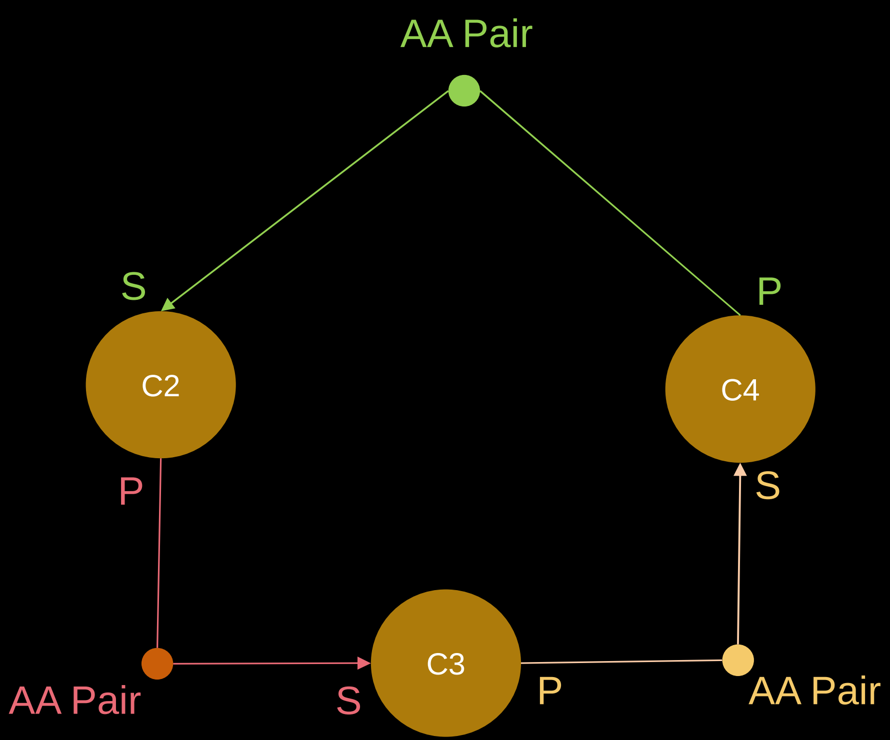


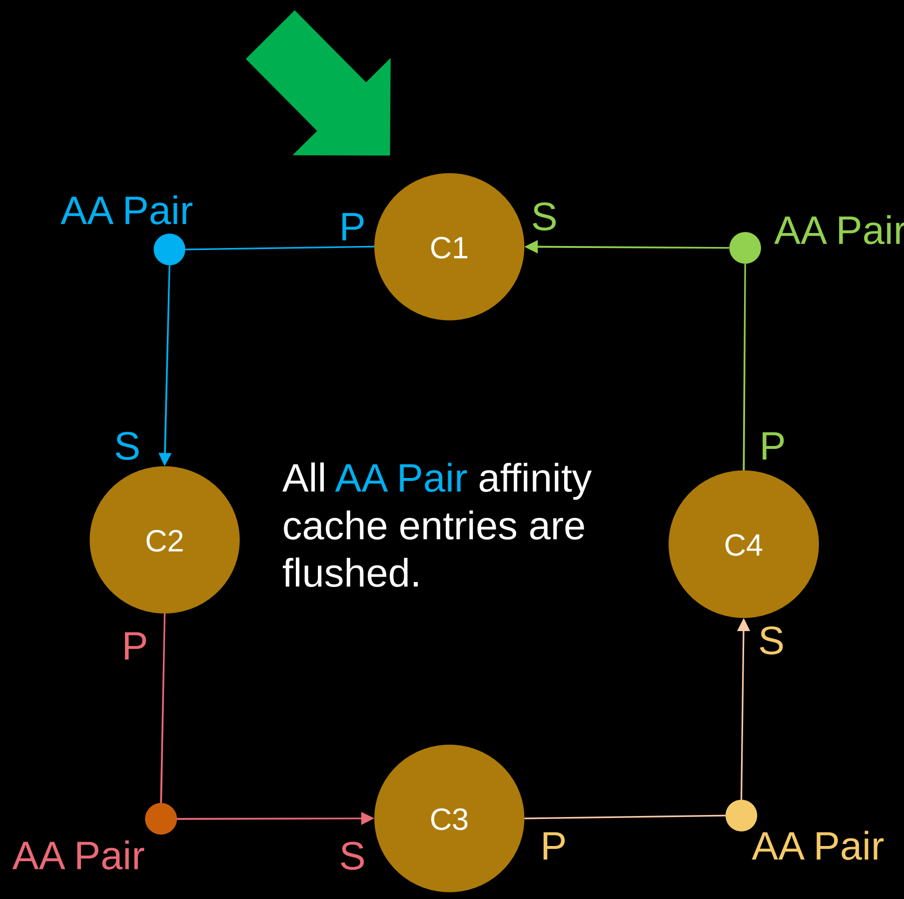
- **For a non-id specific request the proxy will send it to the least loaded AA pair.**
- **For an id specific request where the id is in the affinity cache, the cached AAPair is used**
- **For an id specific request where the id isn't in the affinity cache the least loaded AA pair will be selected and that selection cached.**
- **For all requests going to an AA pair no matter how they got there**
  - The AA pair will send it to the primary (P) core
  - The AA pair will immediately send it to the secondary (S) core.
- **Core loading is based on the responses since only one of the 2 cores in a pair will process the request.**

Active-Active Core Removal

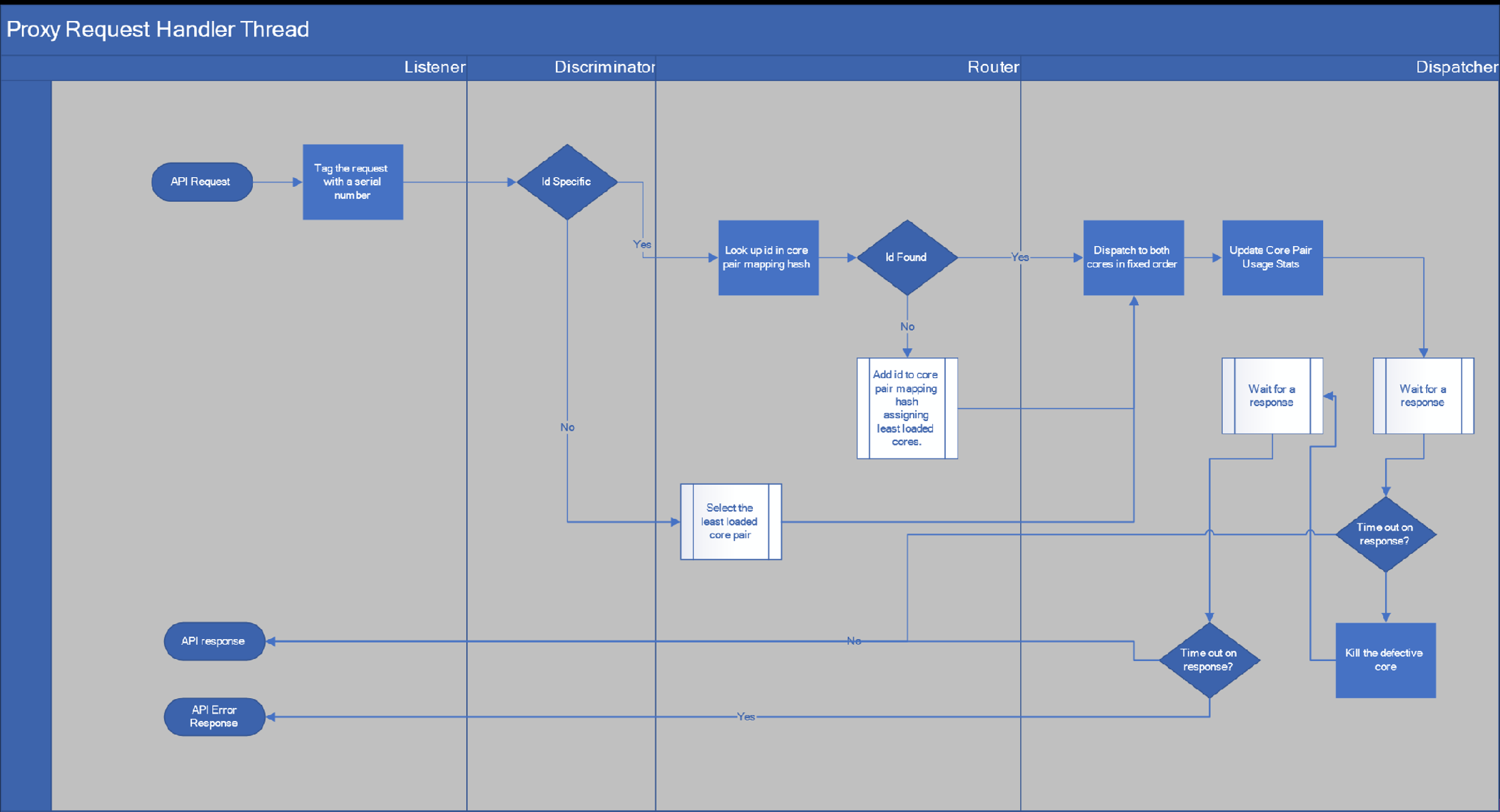


Active-Active Core Addition

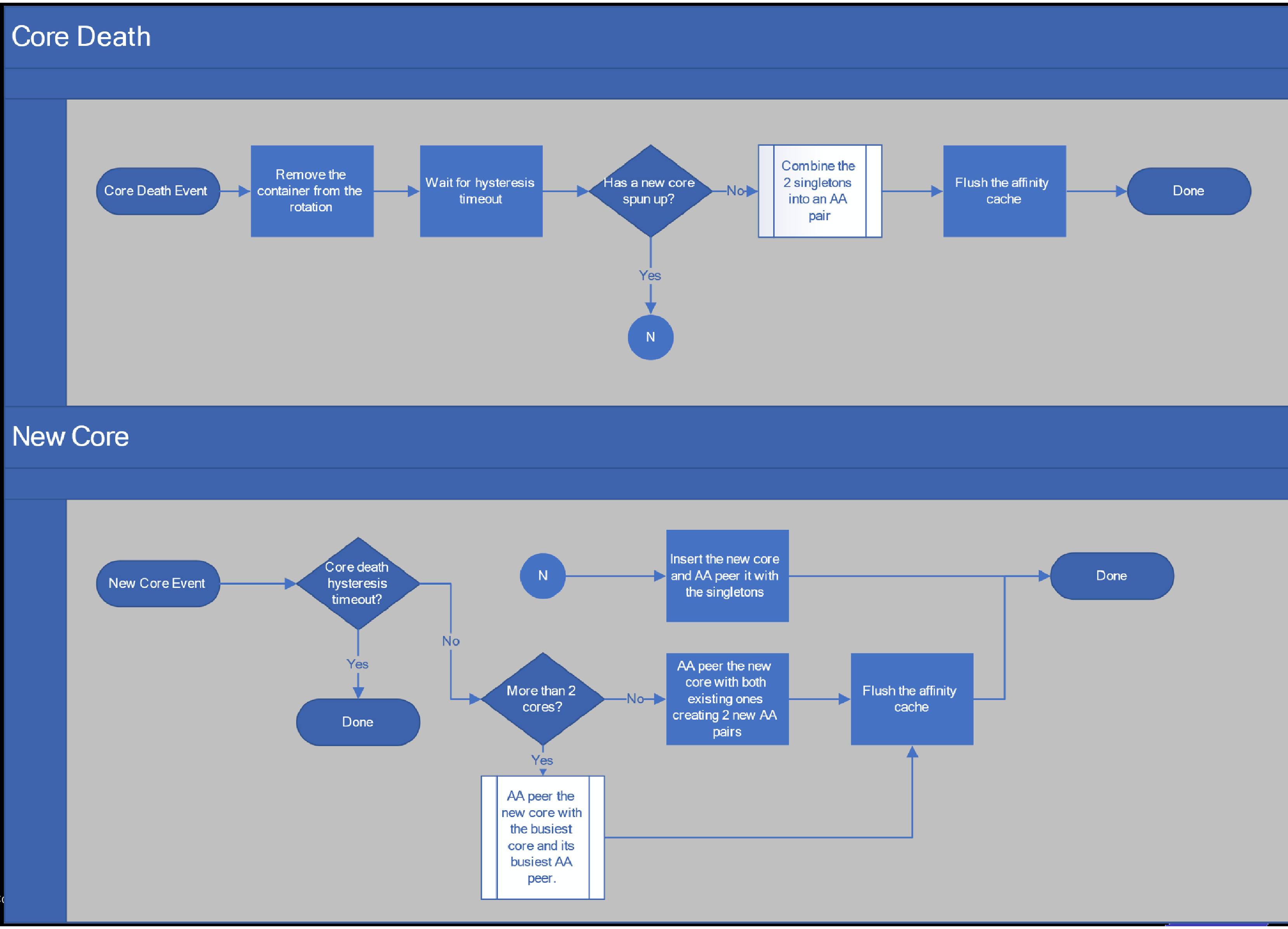




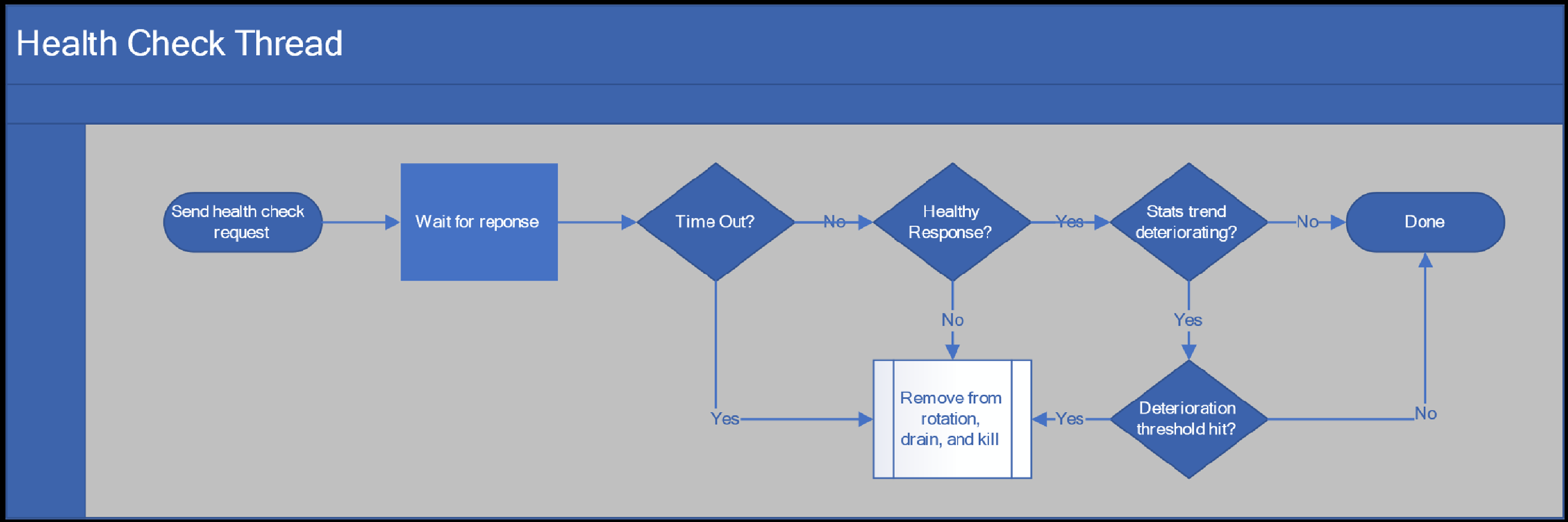
Algorithms / Flowcharts



Proxy Request Handler Thread
Listener
Discriminator
Router
Dispatcher
API Request
Tag the request with a serial number
Id Specific
Yes
No
Look up id in core pair mapping hash
Id Found
Yes
No
Dispatch to both cores in fixed order
Update Core Pair Usage Stats
Add id to core pair mapping hash assigning least loaded cores.
Wait for a response
Wait for a response
Select the least loaded core pair
Time out on response?
API response
No
Time out on response?
Kill the defective core
API Error Response
Yes

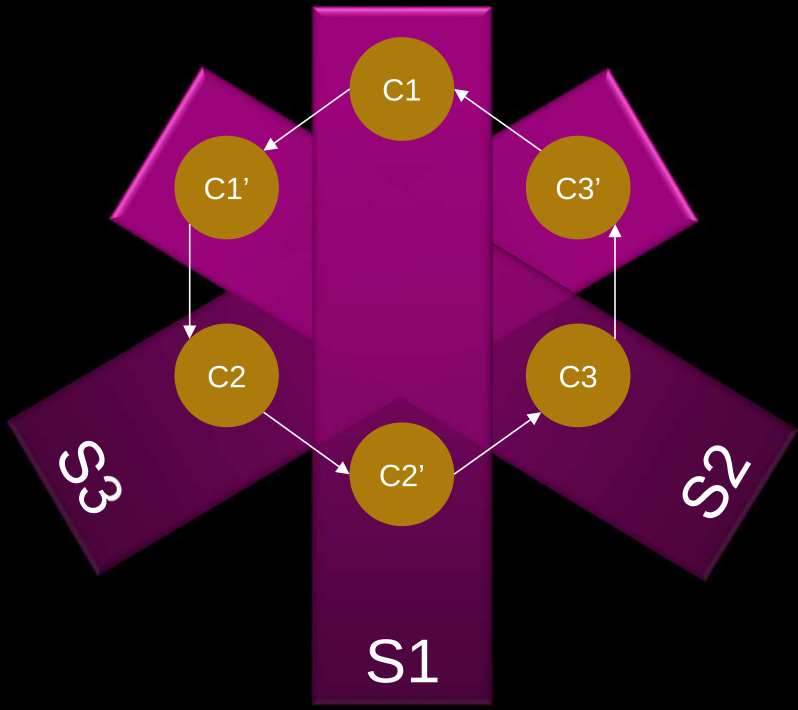


Core Death
Core Death Event
Remove the container from the rotation
Wait for hysteresis timeout
Has a new core spun up?
No
Combine the 2 singletons into an AA pair
Flush the affinity cache
Done
Yes
N
New Core
New Core Event
Core death hysteresis timeout?
Yes
Done
No
More than 2 cores?
No
AA peer the new core with both existing ones creating 2 new AA pairs
Flush the affinity cache
Yes
AA peer the new core with the busiest core and its busiest AA peer.
N
Insert the new core and AA peer it with the singletons
Done


Health Check Thread
Send health check request
Wait for reponse
Time Out?
No
Healthy Response?
Yes
Stats trend deteriorating?
No
Done
No
Yes
Yes
Remove from rotation, drain, and kill
Yes
Deterioration threshold hit?
No

Core distribution across HW

Core distribution across HW

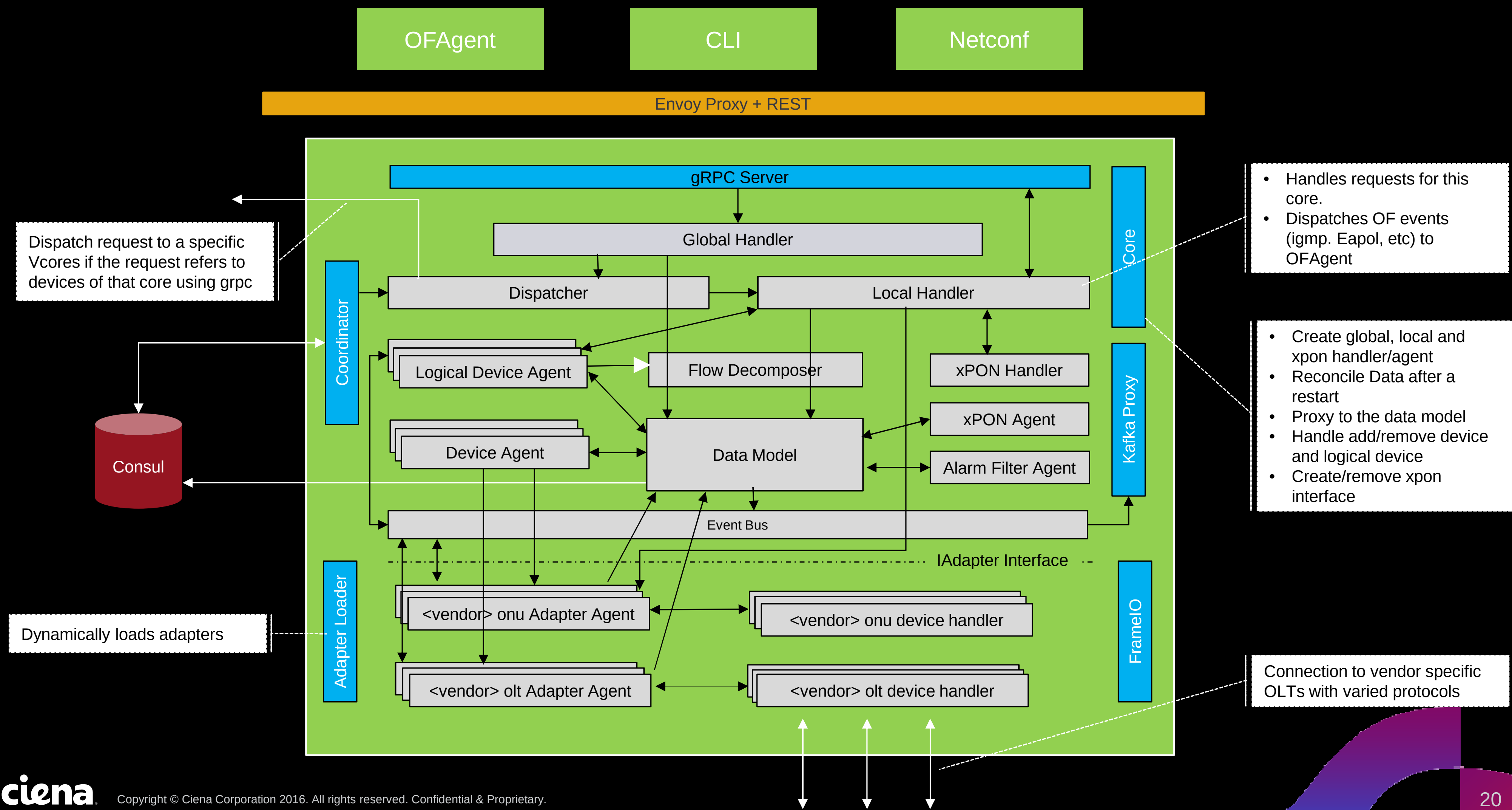


- **One possible core distribution is shown on the left.**
- **This is the minimum HA configuration, 3 servers and 6 cores.**
- **A core and it's peer should never be co-located on the same server**
- **For better load distributions, each server should have a non-prime and a prime not associated with the non-prime.**
- **Shown is S1:{C1,C2'}, S2:{C1',C3}, S3:{C2,C3'}**
- **Could have also distributed as: S1:{C1, C2}, S2{C2',C3'}, S3{C3,C1'}**
  - this would have been less optimal since the non=prime cores are expected to be the busiest based the proxy behavior.

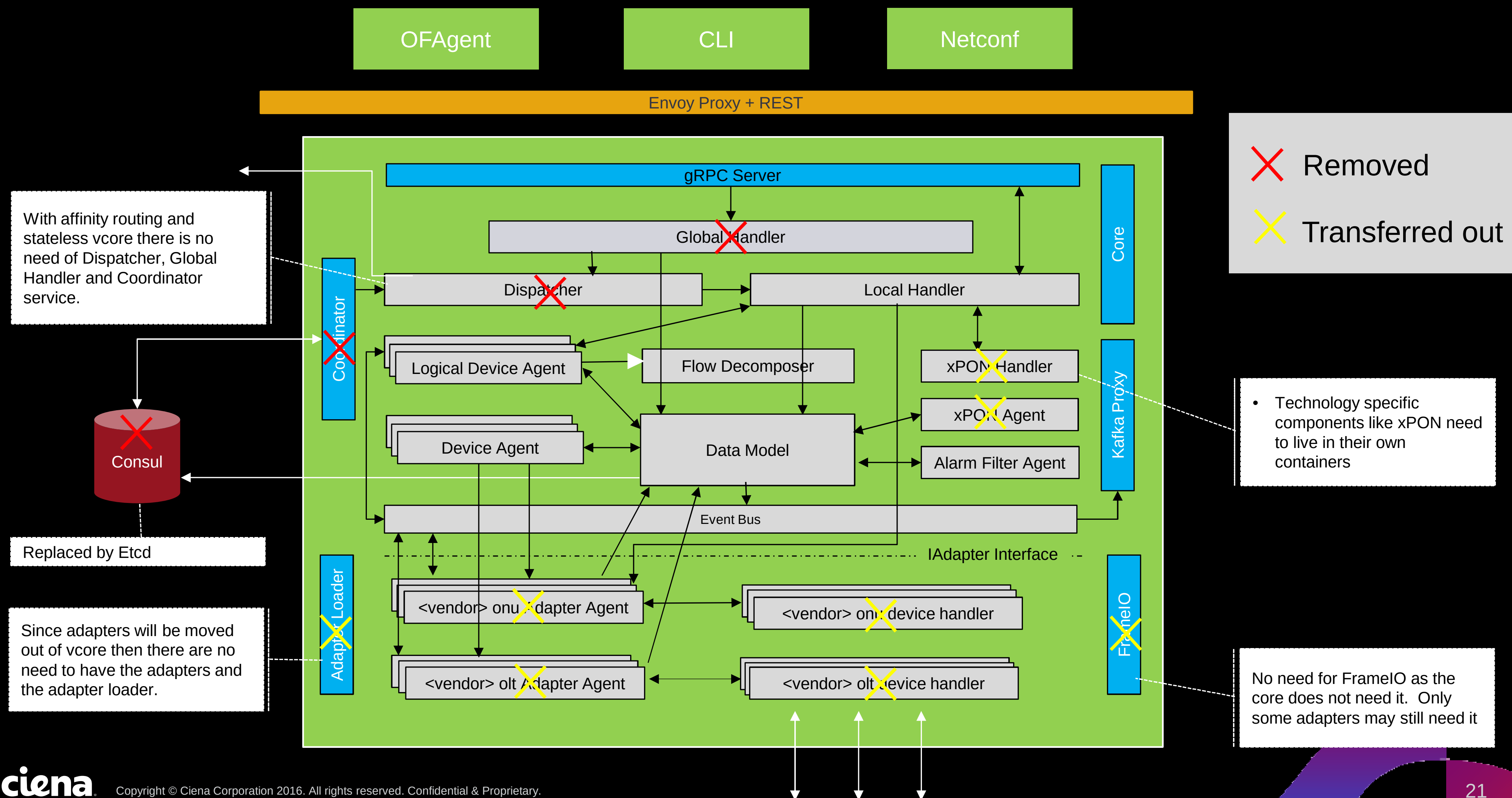
Voltha Core Design

Current Core Overview

Voltha Core - Current Design

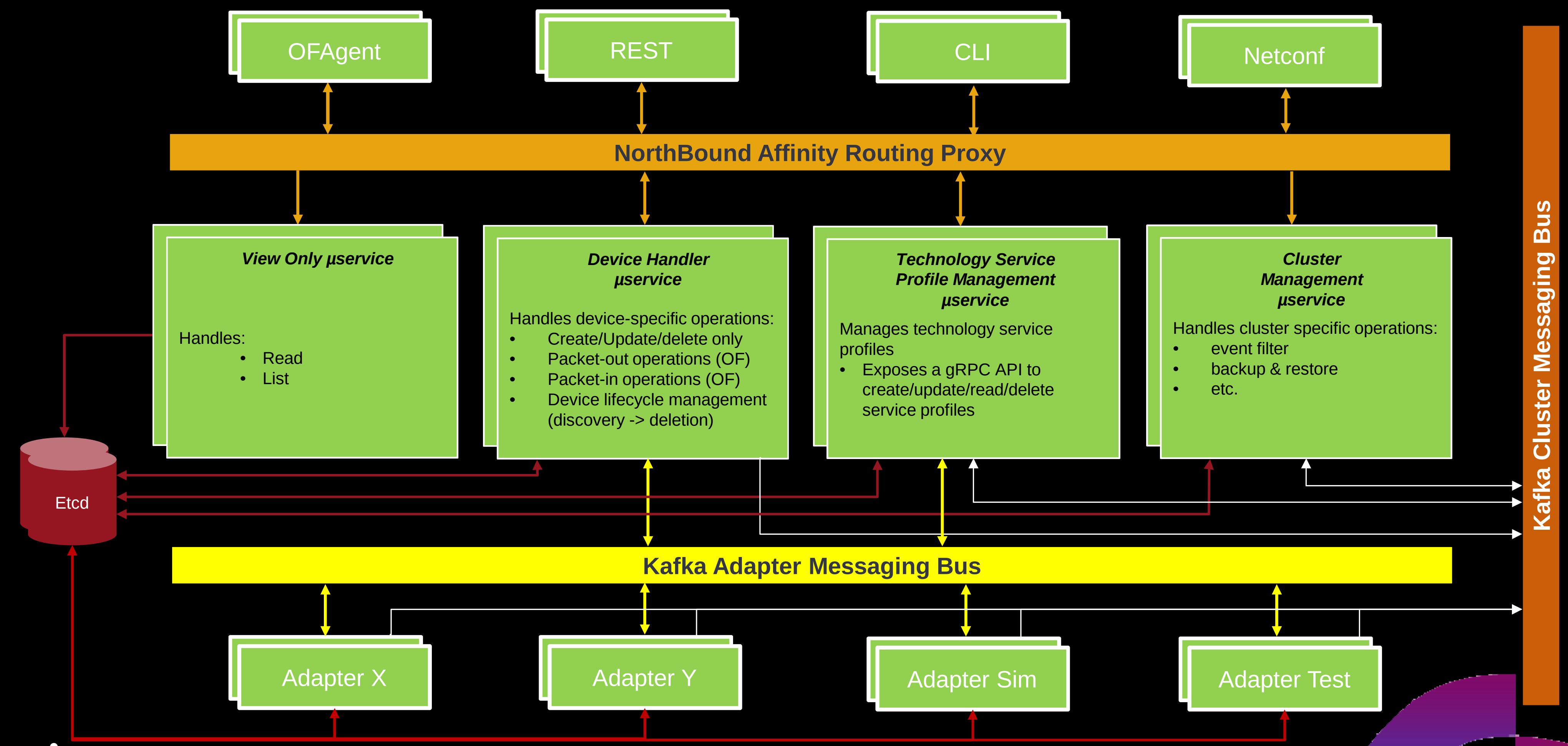


Voltha Core 2.0 – Removed/Transferred out components

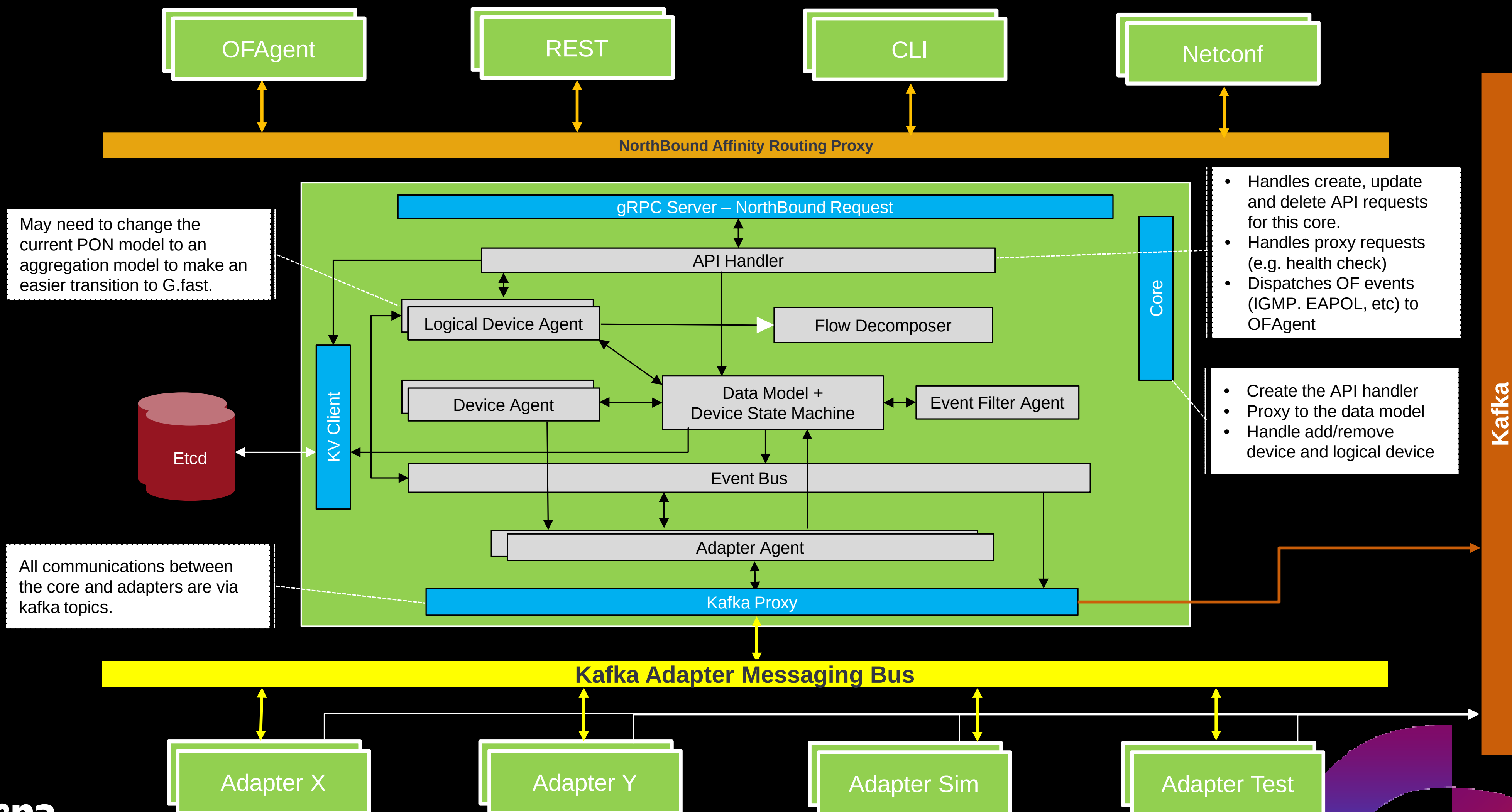


Proposed Core Architecture

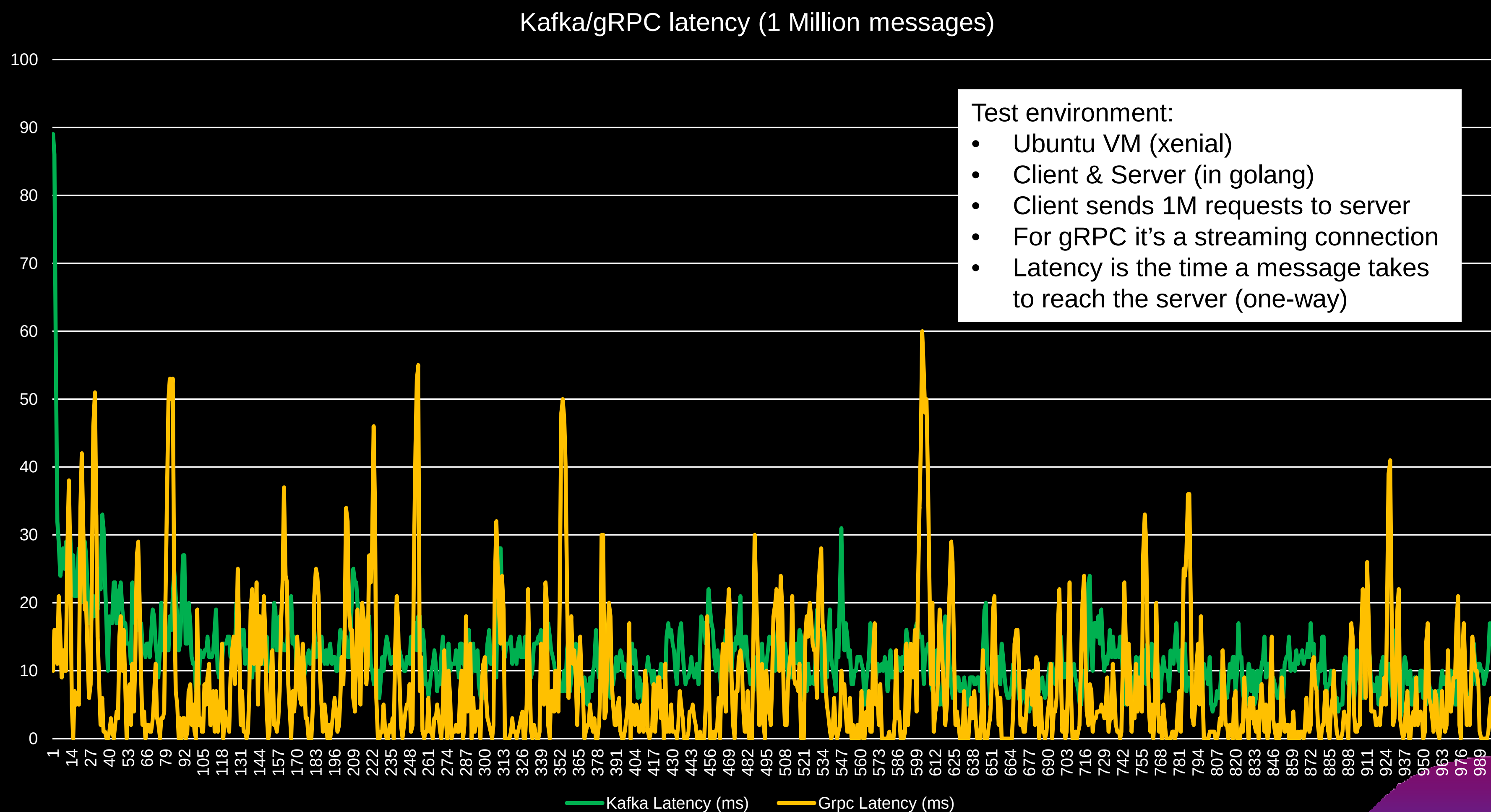
Voltha Core 2.0 – 4 µservices



Device Handler µservice



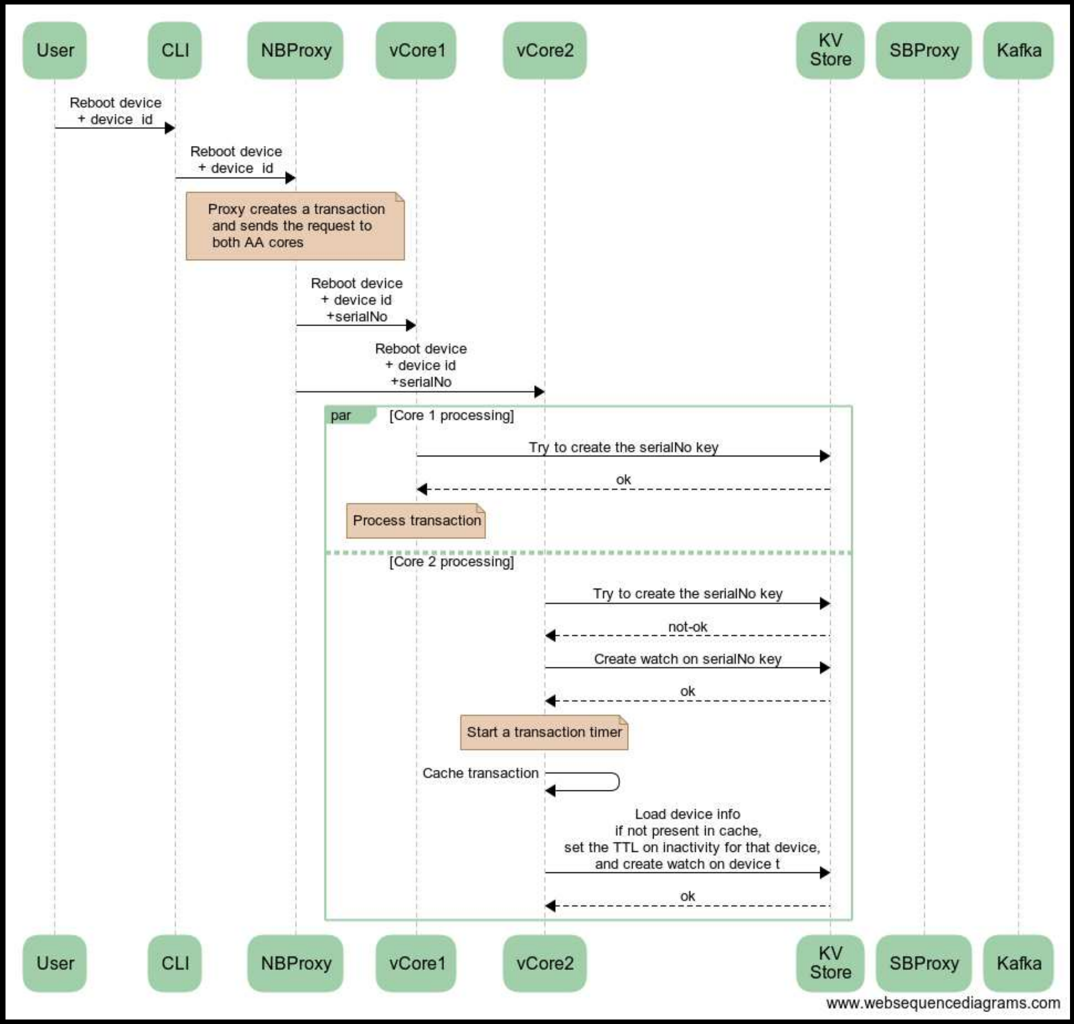
Message latency – Kafka vs gRPC



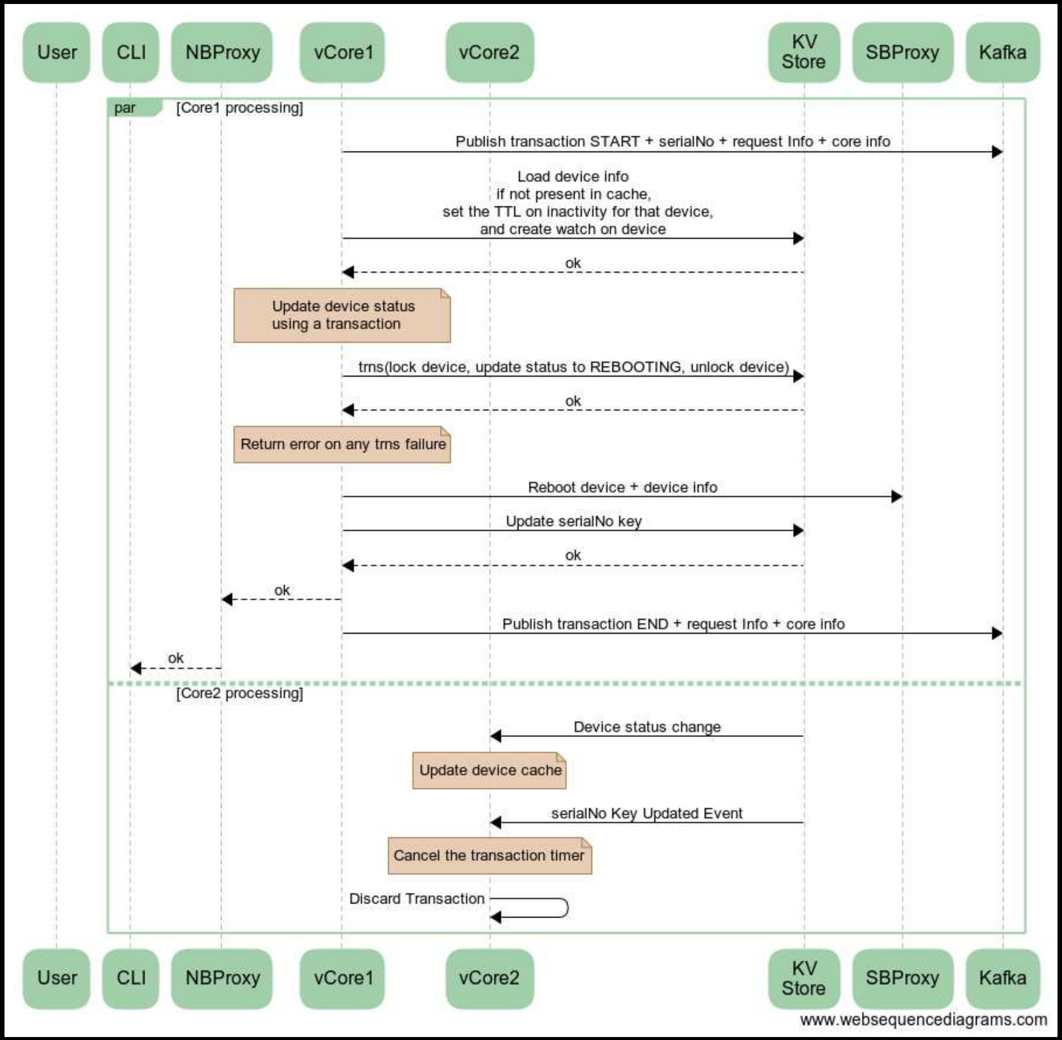
Test environment:
- Ubuntu VM (xenial)
- Client & Server (in golang)
- Client sends 1M requests to server
- For gRPC it’s a streaming connection
- Latency is the time a message takes to reach the server (one-way)

Success & Failure Examples

Example– Reboot request (no error) – part I

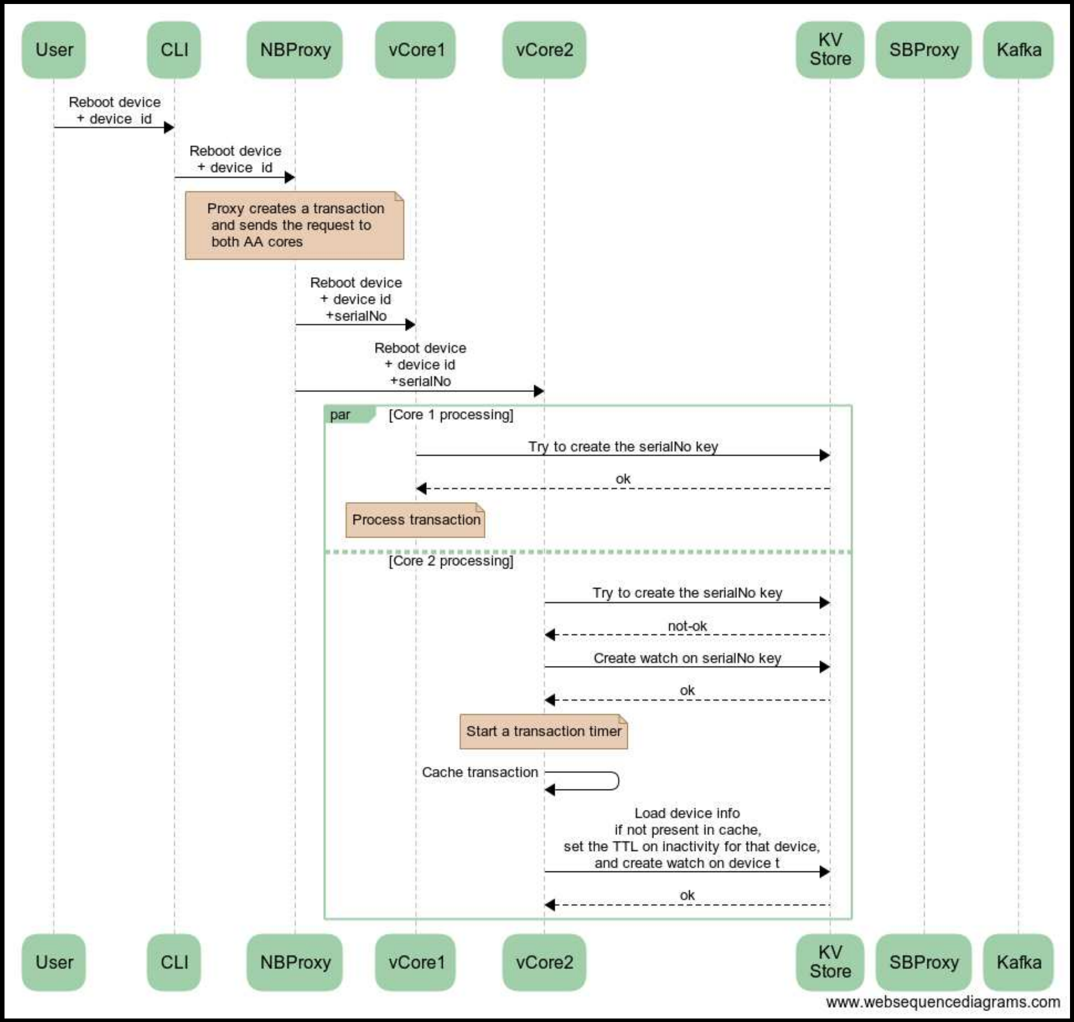


Example– Reboot request (no error) – part II



ciena

Example– Reboot request (error) – part I



Example– Reboot request (error) – part II