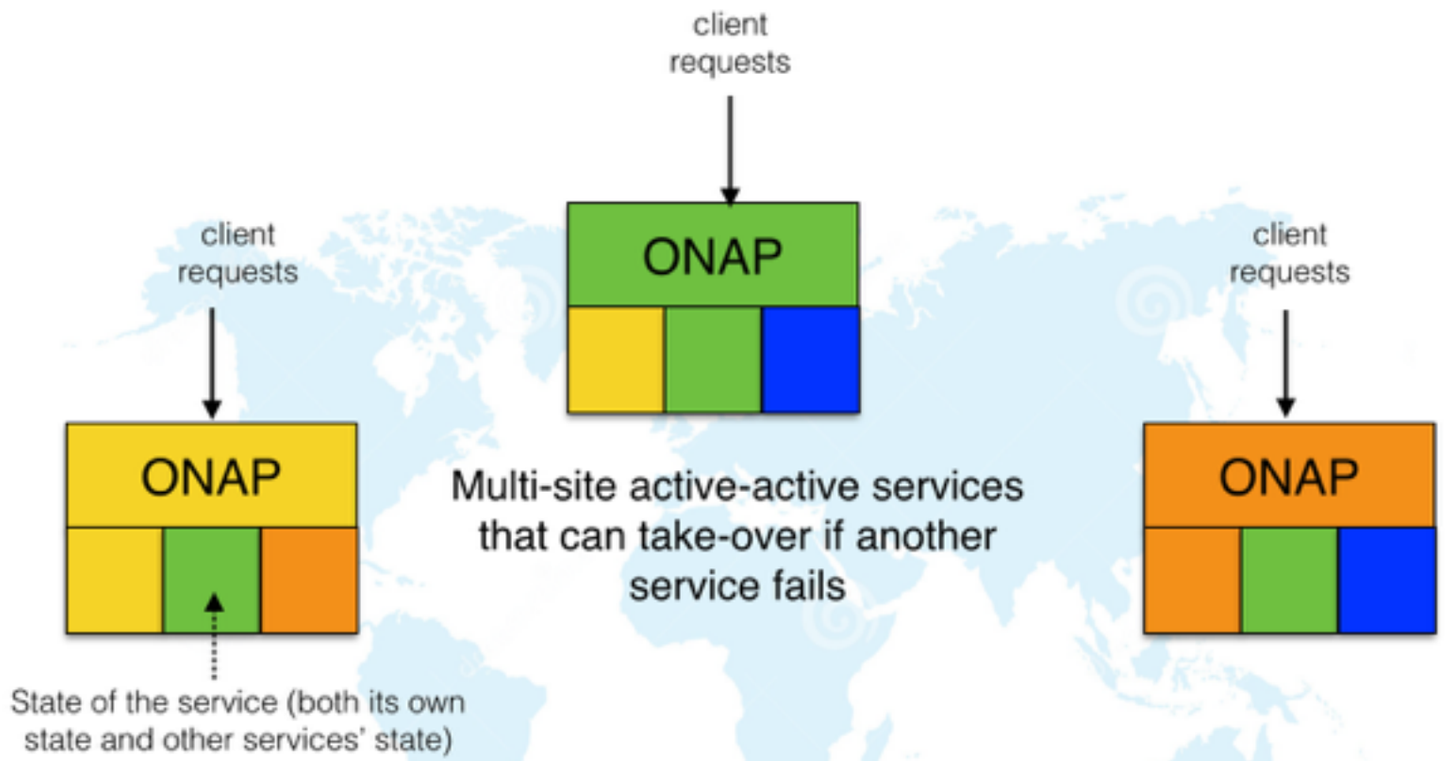
Common High-Availability Platform (CHAP) for ONAP Resiliency

Bharath Balasubramanian*, Michael Howe** and Rick Schlichting*
*+Network Cloud Infrastructure, **Network and Shared Services Development

Goal: A common high-availability platform to build ONAP components with 5 9s of availability on 3 9s (or lower) software and infrastructure in a cost-effective manner.

To achieve this goal ONAP components need to support multi-site, active-active services with efficient failover.



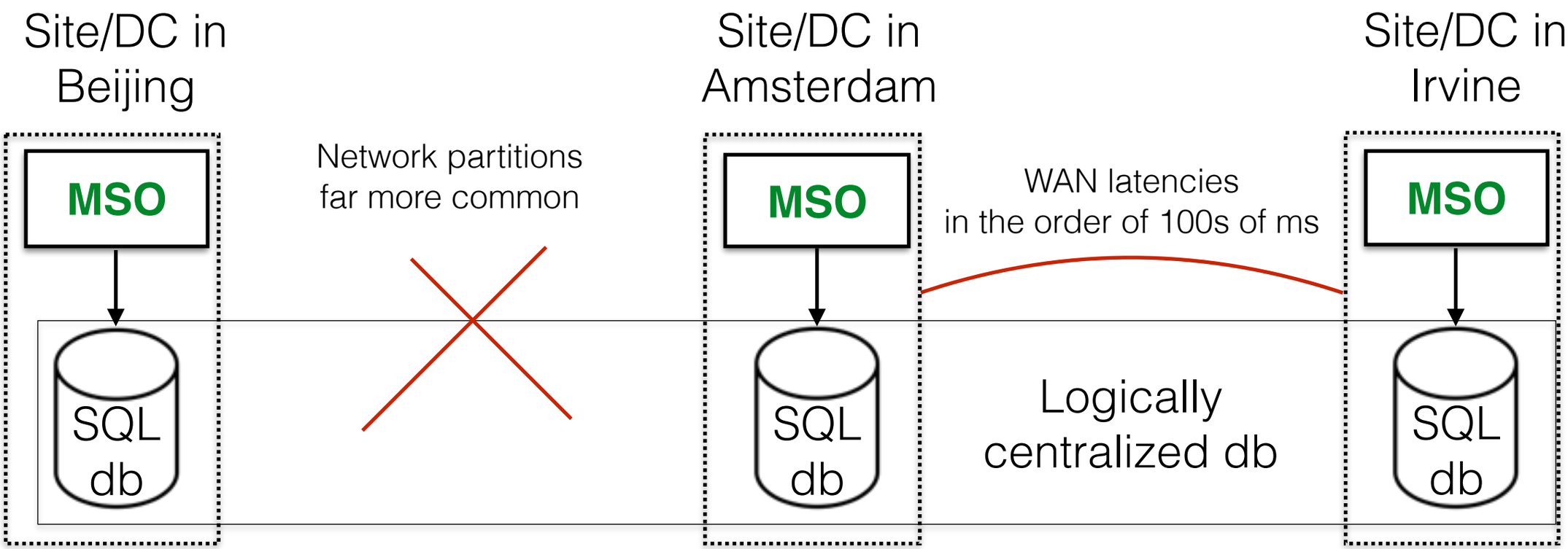
Problems

Deploying replicated components across multiple geo-distributed sites — need to deal with *WAN latencies* and *network partitions*.

Active-Active, failover architecture — need to design *complex distributed protocols* for failure detection, federation, leader election.

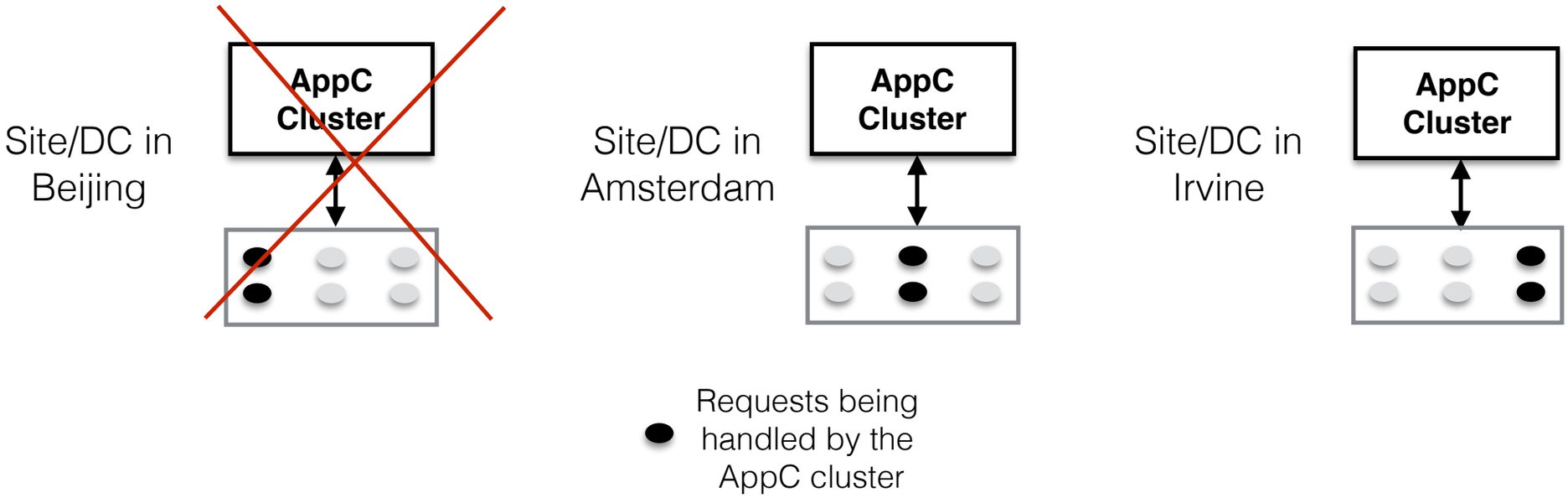
Diverse replication and resiliency requirements, e.g. some ONAP components have replicated state, while some don't. Some need inputs to be processed by just one replica, others don't care — *cannot have a one-size-fits-all* replication/resiliency solution.

E.g. Multi-site state replication



Clustered, transactional DB solutions like Maria-DB may not satisfy *latency* and *availability* needs across geo-distributed sites!

E.g Complex distributed protocols



- How can failure of an AppC cluster be detected in a correct manner?
- Where and how should the existing requests be reallocated?
- How can we ensure no split-brain problems in case of erroneous failure detection?

E.g. Diverse Replication Requirements

SDN-C state for most parts can be replicated across sites in a lazy manner — may not need strong replication semantics.

AppC requests may be processed by only one cluster — need strong replication semantics.

Some MSO-Camunda tables seem critical for state management, while some seem to be just for record keeping — need mixed replication semantics.

Current Practice

Each team building its own solution — wasteful and can often be erroneous due to complex distributed protocols, *replete with corner cases*.

E.g.selective database mirroring techniques, failover using pacemaker+corosync, fault-detection loosely based on timeouts.

Proposed Requirement: CHAP

- Identify common availability and resiliency problems across ONAP.
- Provide a common high-availability platform (CHAP) with shared services.
- ONAP components can simply configure and use CHAP according to their needs.

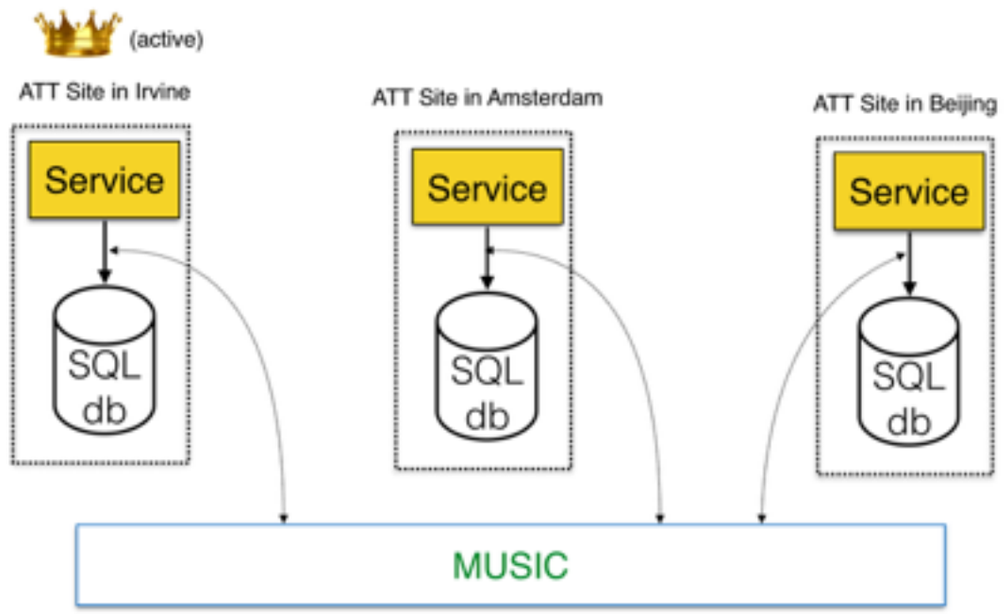
CHAP Building Blocks

- MUSIC: Multi-site geo-distributed database for state-management at scale.
- HAL: Configurable recipes for complex federation and resiliency protocols.
- Conductor: Site-selection service to satisfy diverse service constraints on resiliency and availability during initial placement and failover.

MUSIC for multi-site state management

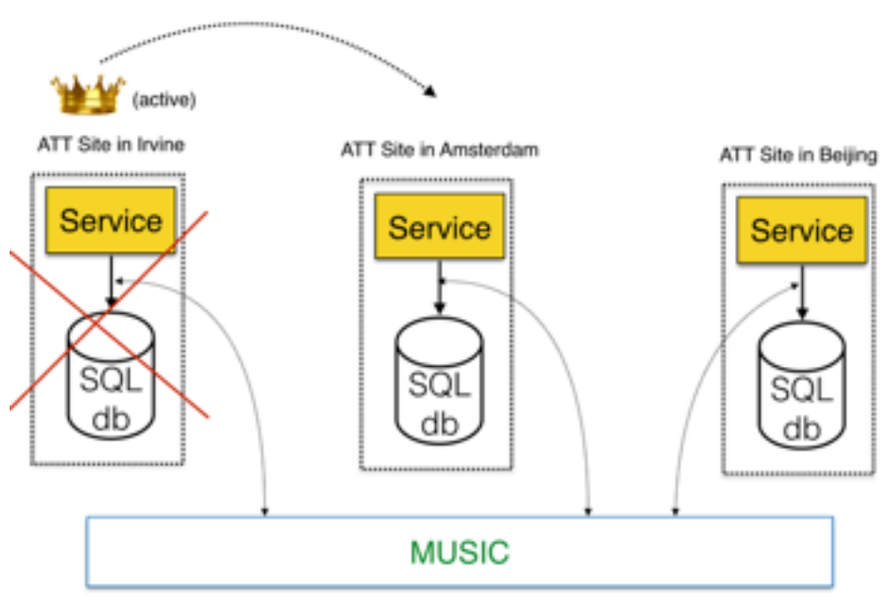
MUSIC (multi-site coordination) is a geo-distributed state-management service, where:

- The Active service reads and writes to a high performance local SQL db
- This state can be synchronously/asynchronously replicated to the standby.
- When a standby has to take over on active failure, MUSIC provides it with the latest state.



HAL for high-availability recipes

Detecting failure of a service and transferring control to a standby on another site involves complex distributed system challenges, replete with corner cases.



HAL provides recipes on top of MUSIC that services can simply configure to achieve different resiliency patterns: active-standby, active-active with failover etc.

Conductor for Site-selection

Services often have complex, diverse constraints on how client requests should be routed to service replicas:
e.g. “Requests must be sent to replicas within 10 ms latency that are less than 50% utilized”

Conductor provides a configurable constraint solver that matches requests to sites/replicas based on individual service policies — used during both initial request assignment and failover.

Tool/Code complexity

- MUSIC uses two production tested tools Cassandra (to store state) and Zookeeper (for locks) without any modifications with just around 2000 lines of an additional shim to tie them together — nearly two years of testing and deployment within ATT.
- HAL uses Zookeeper locks (can be Consul/etcd) for leader-election and simply uses MUSIC to store heart-beats — less than 1000 lines of code.
- The constraint solver in Conductor is less than 1000 lines of code — nearly two years of testing and development within ATT.

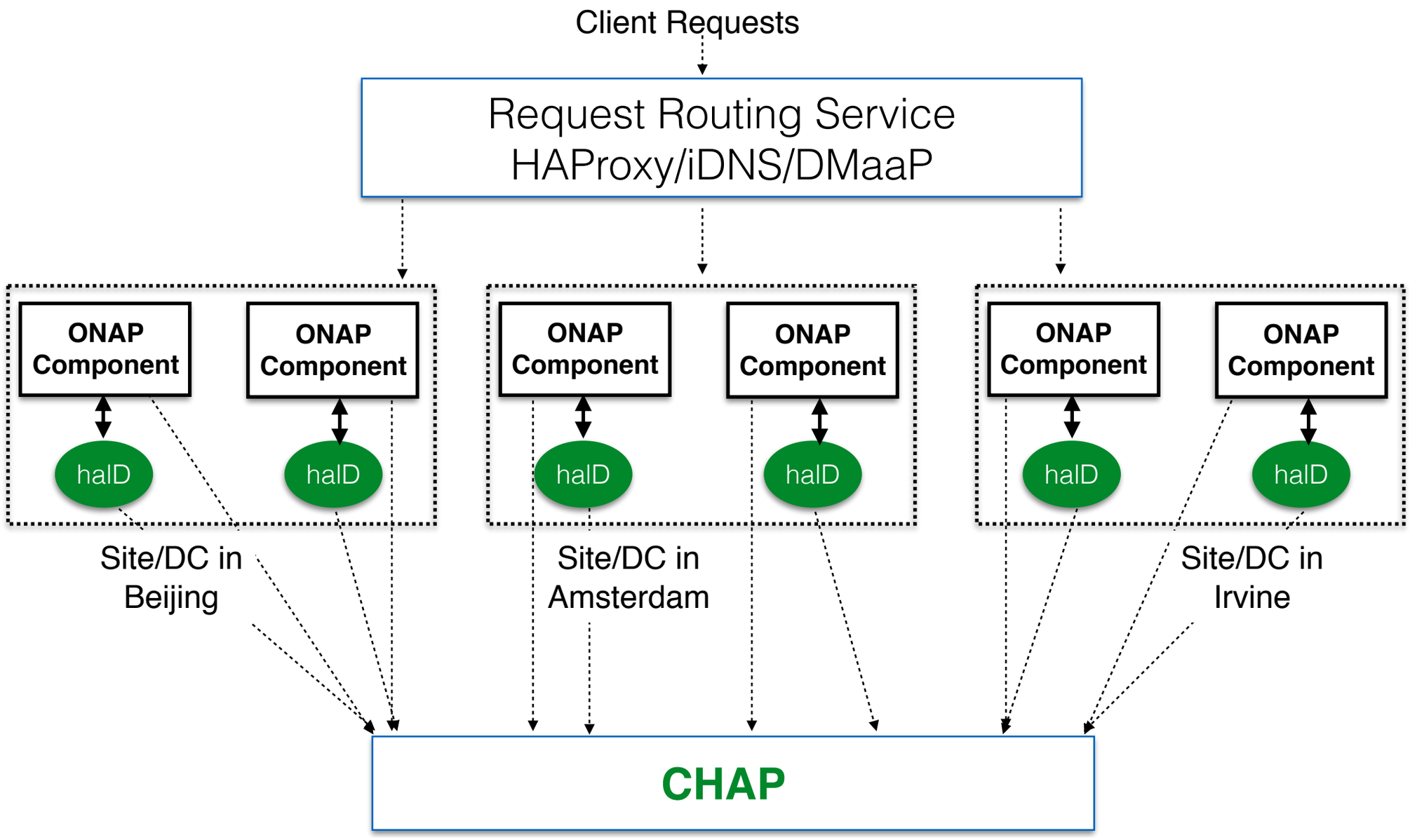
ONAP-CHAP Design with ATT TechDev Partners

Common ONAP resiliency design pattern

The most common resiliency pattern that ONAP components seems to be targeting:

- three sites, each hosting two ONAP component replicas (OCR for short)
- Requests are partitioned across the OCRs (active-active system)
- If an OCR fails, then its requests must be taken over by the site-level OCR
- If the entire site fails then the requests must be distributed across the remaining OCRs as appropriate
- To ensure efficient failover, OCR state that is maintained in a database should be replicated across sites

ONAP CHAP Architecture



CHAP platform providing Conductor (distributed site selection service) that is used by the halDs and MUSIC (distributed multi-site state management service) that is used by both the halDs and ONAP components

Key Elements

* the green elements are part of CHAP

- The **ONAP component replicas (OCRs)** each of which process client requests. E.g. MSO replicas, AppC replicas. The ONAP components can write their state to MUSIC/mdbc directly or through the HAL daemon (as shown in this example).

- A request routing service (RRS) that directs client requests to the appropriate OCR. This does not do the job of selecting the OCR (which is done by Conductor done below).

- The **HAL daemon (halD)** is a light-weight stateless representative of CHAP that runs as a companion to each ONAP component replica (typically in the same failure domain). The HAL daemon performs distributed failover that includes failure-detection (for crash, partition faults), replica selection and leader election.

- The **Conductor** service that takes as input a client request (s) and details regarding an ONAP component (like current load) and returns a YES if the particular ONAP component can process that request, based on optimization criteria and constraints that can be configured.

- The **MUSIC** distributed state persistence/coordination service that is used by Conductor, each HAL daemon and the ONAP component for state management/inter replica communication etc.

Failure-free flow

1. All the ONAP component replicas (OCRs) are started along with their companion **HAL** daemons across a certain number of distributed sites (typically three).

2. When a new unassigned client request arrives, the RRS simply sends to all OCRs that independently asks its companion **HAL** daemon if it should process this request.

3. The **HAL** daemon queries **Conductor** with the details of the request and details of its companion OCR (like load) and **Conductor** replies with a YES/NO. **NOTE**: each request need not go through Conductor and can be aggregated (like topic in AppC) so that all requests corresponding to a certain aggregation level go to the same OCR.

4. All the **HAL** daemons for whom Conductor replied with a YES for this particular request will perform leader/primary election and only one will win. That HAL daemon will return YES to its companion OCR.

5. The OCR that gets a YES from its companion **HAL** daemon will tell the RRS that it is responsible for that request and start processing the request.

6. The OCR and its companion **HAL** daemon periodically check and ensure that each other is alive.

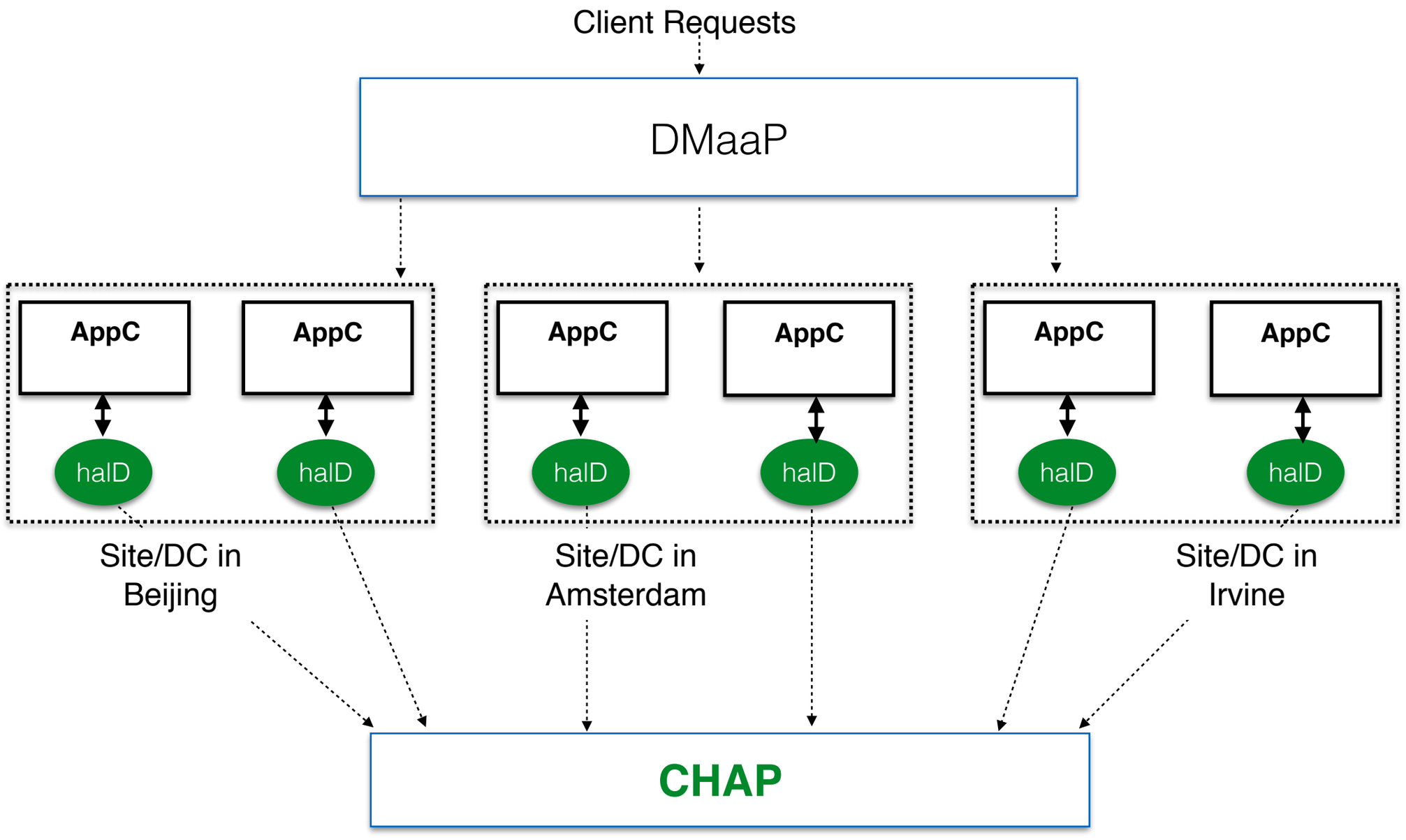
7. Each OCR periodically writes its state to **MUSIC**. This is replicated across the sites (no. of replicas is configurable).

Failover flow

1. An OCR and its companion **HAL** daemon fail (say the VM or even site on which they are running).

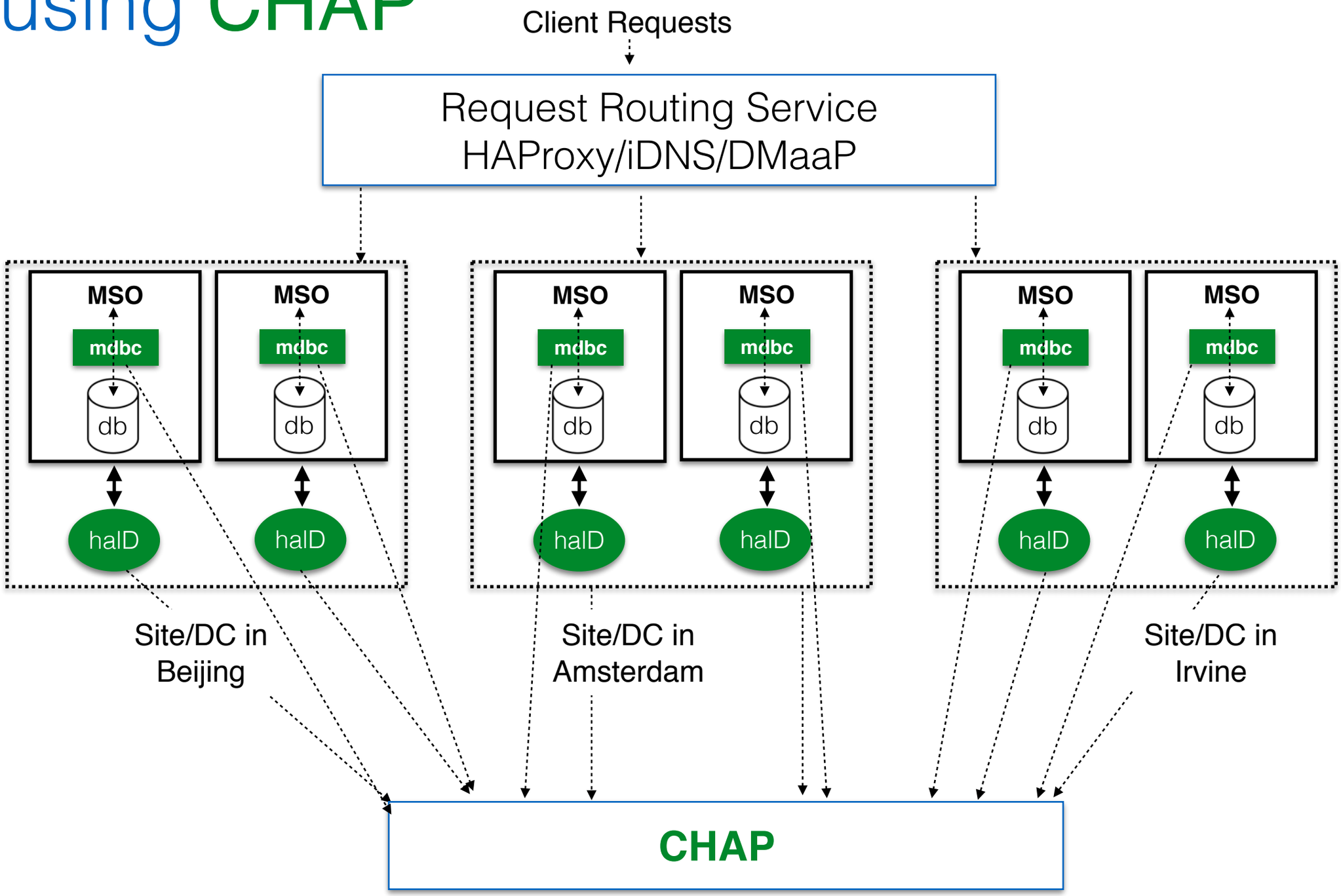
2. The other **HAL** daemons detect this failure and each of them perform steps 3-7 of the previous slide. **Conductor** can be configured to prioritize site-level failovers before choosing OCRs from another site.

Example 1: Stateless, Active-Active AppC using CHAP



CHAP platform providing Conductor and and MUSIC that are used by the halDs.

Example 2: Stateful, Active-Active MSO using CHAP



CHAP platform providing Conductor that is used by the halDs and MUSIC that is used by the halDs and by the thin mdbc layer that *transparently* intercepts the MSO calls to its local SQL database.

Points to consider..

- CHAP can be adapted to other design patterns, e.g. 3 replicas in one site, active-passive systems, etc.
- CHAP can also be used for load-shedding and load-balancing during operation, e.g. use the HAL daemon to detect and transfer load
- Its MUSIC support different databases for the ONAP components, e.g. Cassandra, H2, MariaDb etc.
- All the tools involved have been tested and open sourced

Thank you!

CHAP: A common high-availability platform to build ONAP components with 5 9s of availability on 3 9s (or lower) cloud infrastructure in a cost-effective manner.

- MUSIC: Multi-site geo-distributed database for state-management at scale.
- HAL: Configurable recipes for complex federation and resiliency protocols.
- Conductor: Site-selection service to satisfy diverse service constraints on resiliency and availability during initial placement and failover.