Real Time Data Event Streaming & Processing (RESP)

ARC Presentation

Habib Madani Habib.Madani@huawei.com
Parviz Yegani Parviz.Yegani@huawei.com

April 24, 2018

Q&A Action Items

Q1: What are the End point Use Cases I & II?

A1: Use Case I is a simple **smart end point** scenario, showing streaming interconnect between publishers(producers) and subscribers(consumers). Use Case II, however, is a full streaming, collection and analytics use case, representing an end-to-end real-time data pipeline for streaming via RESP proposed solution.

Q2: Why asynchronous communication?

A2: The DDS middleware handles asynchronous communications via a backpressure mechanism coupled with the DDS QoS feature configurations. This takes the burden off the application code for handling any retries by enabling an end-to-end RESP-based data processing pipeline with backpressure capability (**DDS<-Akka<-Beam + Flink**).

Q3: What do you mean by real-time?

A3: The DDS middleware enables end-to-end real-time pub/sub streaming for data-centric events from wire protocol to middleware through DDS QoS feature settings (the QoS profile can be configured by taking into account the requirements of the user data (either real-time or near real-time)

Q4: Does DDS support the WAN traffic?

A4: Yes, it supports it very effectively, via the concept of “DDS Links”.

Q5: Does the existing DCAE flow change?

A5: No, this is complementary to the existing ONAP DCAE flows. Design-time and run-time enhancements are made to meet the RESP functional requirements (S7,S11)

Q6: Message guaranteed delivery; how is this handled?

A6: DDS with proper QoS configuration ensures that any lost event to be delivered due to the “Eventual Consistency” capability.

Q7: Does VNF need to be updated?

A7: The VES collector is updated to handle DDS-based real-time streaming (both southbound and northbound). On the southbound a VNF (a sensor, etc) can push data/measurements to the collector which in turn uses DDS to generate a real-time stream to be processed by the analytics engine.

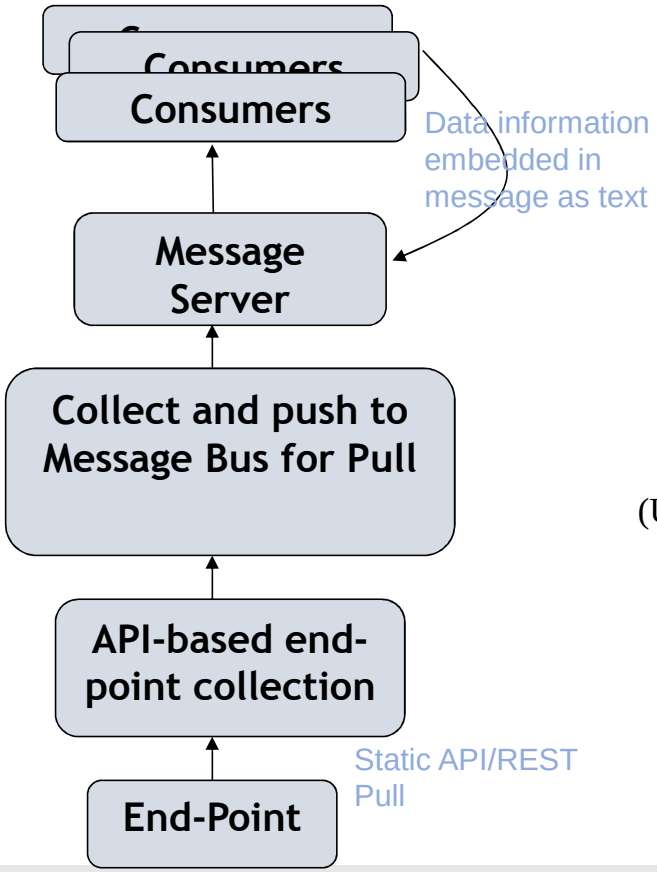
What is RESP?

Real-time data event streaming and processing pipeline comprised of:

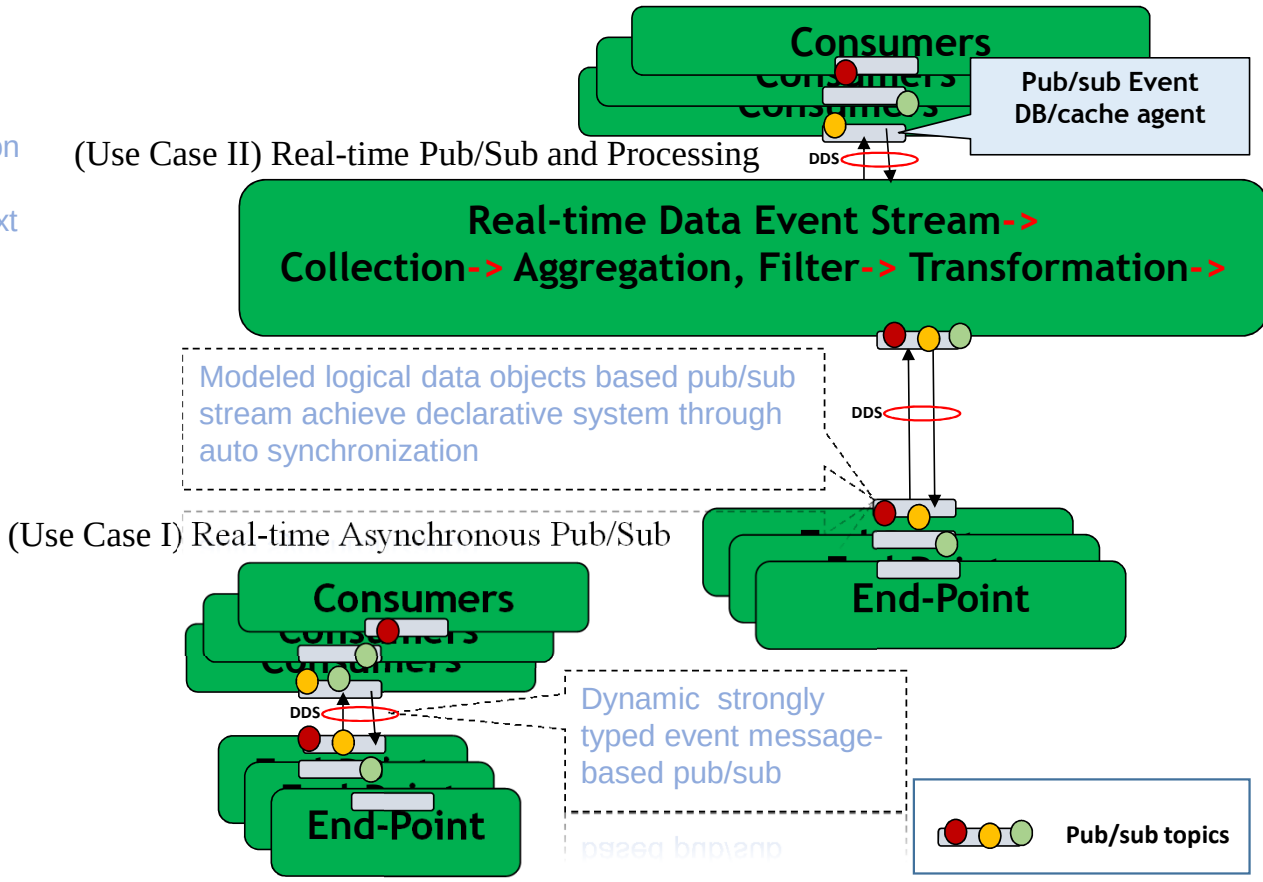
- **Real-time Event Streaming Middleware**
  - QoS-based reliable real-time end-to-end event streaming
  - Non-blocking, highly available collection and routing of events
  - Asynchronous pub/sub based communication
  - Consistent synchronization to accurately represent network states
  - Model-driven approach through data models for consistent structure
- **Real-time Processing for Data Analytics**
  - Efficient and reliable processing of data in motion
    (via reducing a huge amount of data influx through tiered deployment)
  - In-memory very fast, efficient and reliable processing
- **Geo-data Store and Synchronization**
  - Services that need states preserved with consistent geo-data event store and event streaming-based data synchronization

Why RESP?

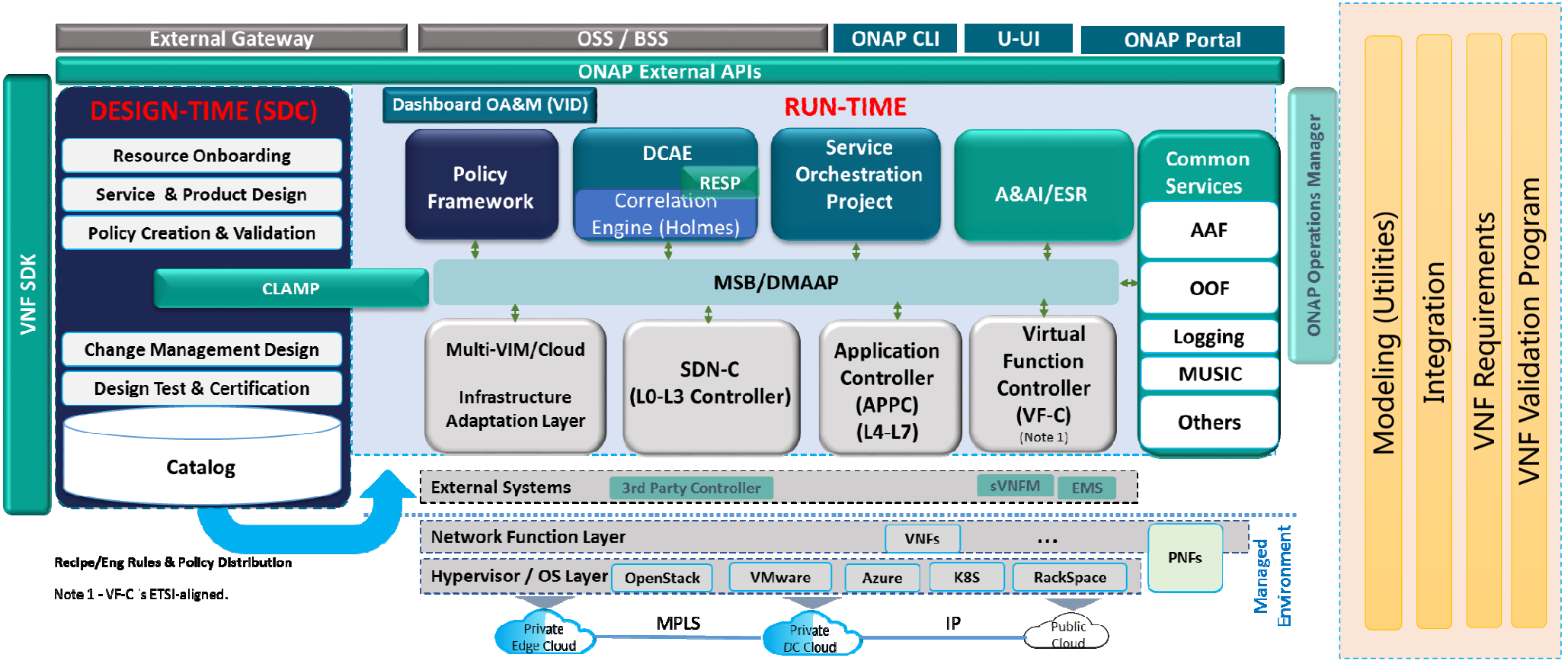
ONAP Current Operation



RESP Pipeline



ONAP Architecture + RESP



RESP Proposal (1/3)

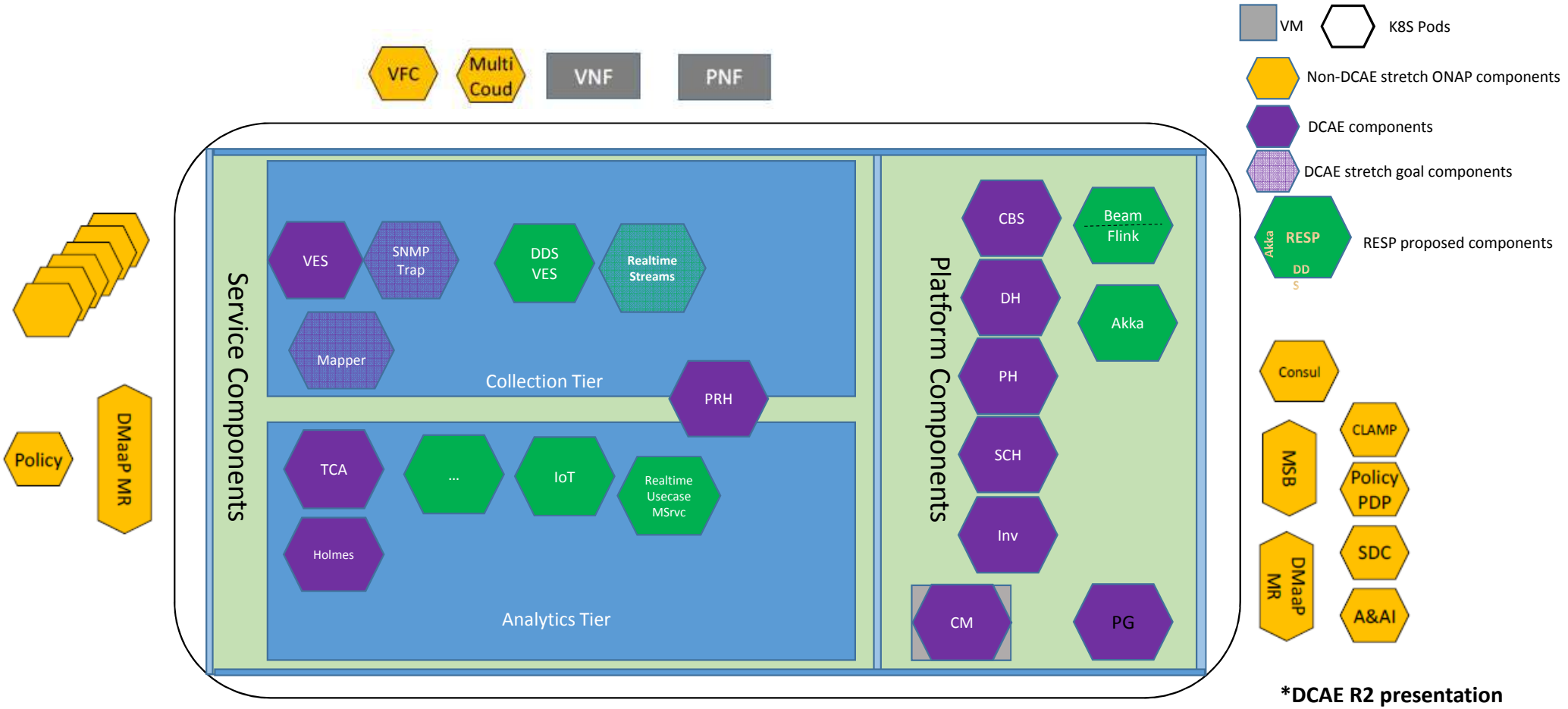
Building on DCAE Base Platform (DCAEGEN3)

DCAEGEN3: Focus on building a platform which is adaptable to changing environment

- **Data Collection:** interfaces keep evolving – Data Streaming, gRPC, etc
- **Analytics:** AI, Machine Learning, Capacity, Cost, Perf., … (real value-add)
- **Common Platform**: 5G (Network Slicing), Wireline, IOT, Cloud, etc
- **ONAP Platform** is used as a base to build from
- **Flink: Real-time event streaming Analytics**
- **DDS: Real time Data centric middleware**
- **Beam: An advanced unified(Batch + Real-time) programing model**
- **Design-Time / Run-Time** separation principles
- **Microservices-**based architecture
- **Data Collector:** collects data from the element (5G network elements, IOT apps,..)
  - Events/traps, statistics, logs, control plane, data plane
- **Data Storage:** collected data will be stored
- **Pipeline Design:** transforms raw data into meaningful data

RESP Proposal (2/3)

Building on DCAE Base Platform (DCAEGEN3)



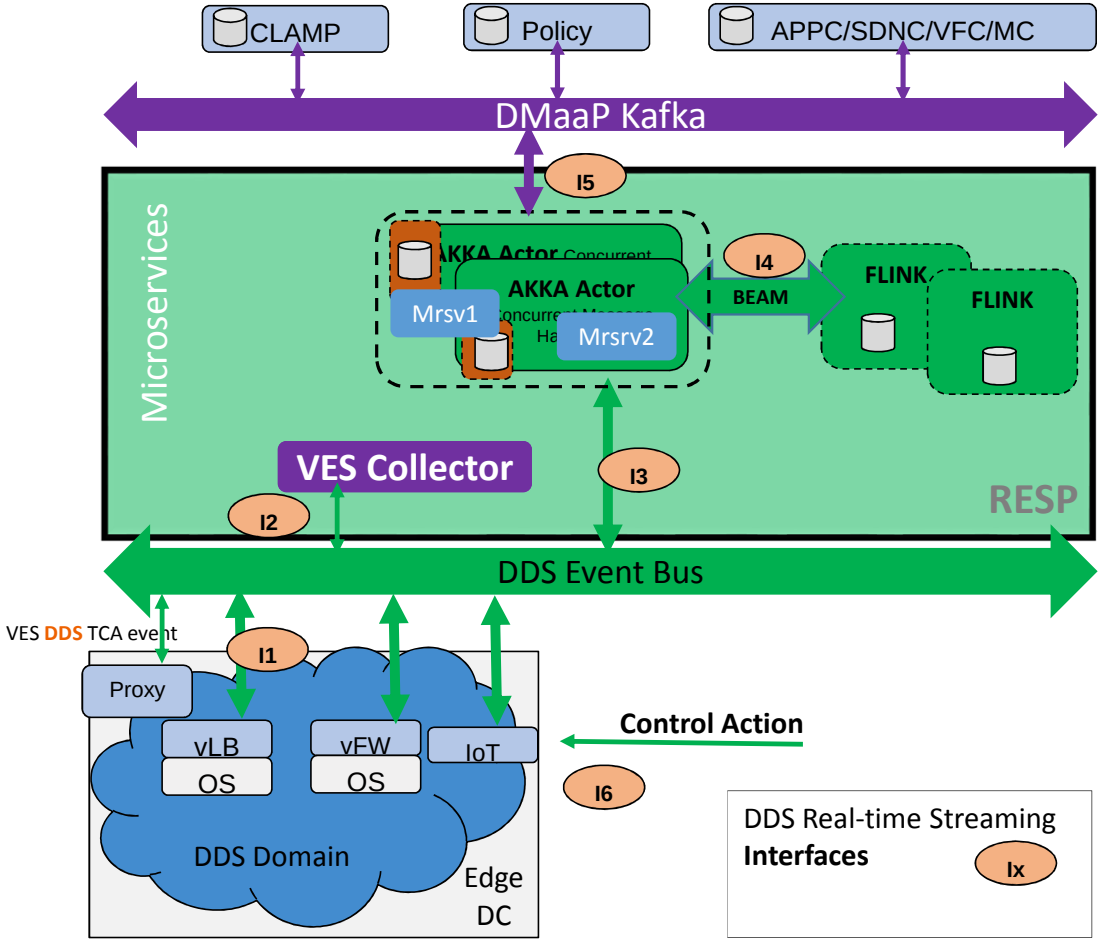
*DCAE R2 presentation

RESP Proposal (3/3)

Building on DCAE Base Platform (DCAEGEN3)

- RESP a new functional entity for DCAEGEN3 Platform
- RESP can co-exist with existing Bus and ONAP components
- RESP impacts ONAP Design-time and Run-time using ONAP APIs (with extensions as necessary)
- All ONAP components potentially can communicate in real-time with common library

NOTE:
- IoT device/VNF can push DDS-based event stream
- Proxy can push DDS-based event streams with legacy backend interface



RESP – Data Analytics and Processing Pipeline

Natively with RESP, mechanisms are provided to

0. Native ONAP APIs leveraged with RESP related metadata and model extensions/additions

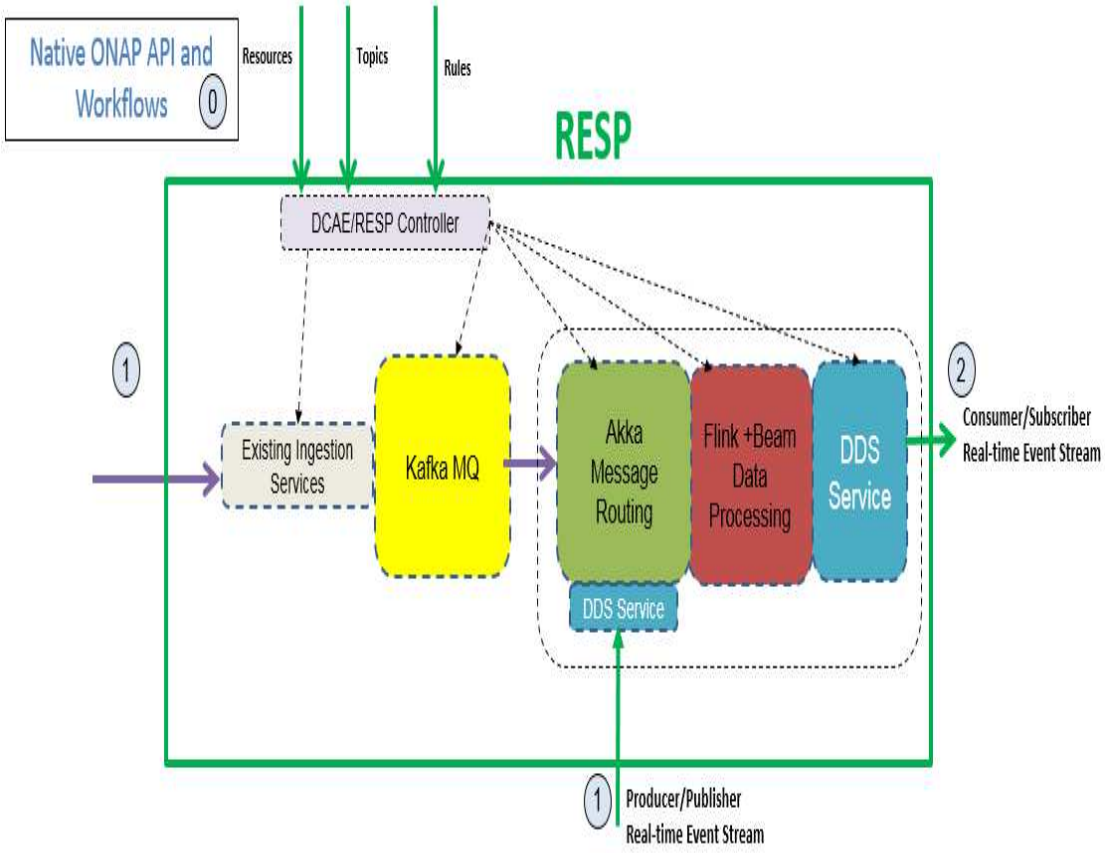
1a. Ingress, **Exactly-Once** and **Eventually Consistent** synchronization mechanism available with DDS-based event streaming between peers.

1b. DDS provides fine-grained control over the **QoS** setting for each of the entities involved in the system.

1c. Ingress and Egress Integration with DMaaP/Kafka Bus and other ingestion needs

- Non-blocking and concurrent routing/handling of data streaming (**Akka**)
- Handle bursty stream processing, in parallel and quickly (**Beam -> Flink**)

2. Egress real-time DDS-based data stream publish



RESP Components

Cloud Native

DDS for Realtime **Data Distribution BUS**

- P2P Tiered Geo Distributed PUB/SUB (Blood Stream)
- Rich QoS support for reliability, extreme low latency support through light wire protocol
- Asynchronous/Synchronous
- Strong Typing(binary), data model/IDL
- Inherent Security
- Discovery, Query and Filter
- Streams/Events/Logs/Payload
- Connectionless optimized
- Publisher/Consumer through a Relationship model
- Reduce Chatter

Akka+Kafka for HA **Collection Framework**

- Real-time Streaming and Batch collectors
- Concurrent, Actor model, Non-blocking
- Distributed, Resilient, parallel processing, Back Pressure
- Collection/Aggregation possible from many streams
- Load Balancing and Partitioning with reliability, HA and *error handling*
- Time Series and Other

Beam & Flink for ***Transformation*** *&* **Analytics Framework**

- **Unified Data Streaming Programming Model and DSL and pipelining**
- Realtime Streaming and Batch Processing
- **Rich Parallel processing with rich API**
- **Resilient Distributed Data processing, with in-memory Cache analytics** for huge stream input
- **ML analytics &Graph/R predictive modeling and analytics**
- **Rich Query and Time Series**
- **Beam allows integration with other runners (Spark, Storm)for compatibility**

DB Persistence **Storage**

- **DDS Cache/Event Store**
- **Need based DB use for backend store**
- **Geo Distributed**
- **Consistency(✓ Eventual) for supporting statefull entities**
- **Backend Connectors for integration of huge K,V store**

RESP Design-time and Run-time Operations

On-boarding new Platform components

Platform Deployment

- DDS VES Collector
- DDS Proxy Collector
- Flink
- Beam enabled MSrvc
- Akka enabled MSrvc

Platform On-boarding of new Templates

Design -Time Deployment

1. Inputs Output Configuration
2. TOSCA model Templates created for Cloudify blueprint
3. Import into SDC
4. Saved in Catalog
5. Data Streaming
    1. VES Event Data formats created

Design Service Creation

Design -Time Deployment

1. DCAE template populate to drive Use Case Microservices (IoT, VoLTE, TCA, Other ..)

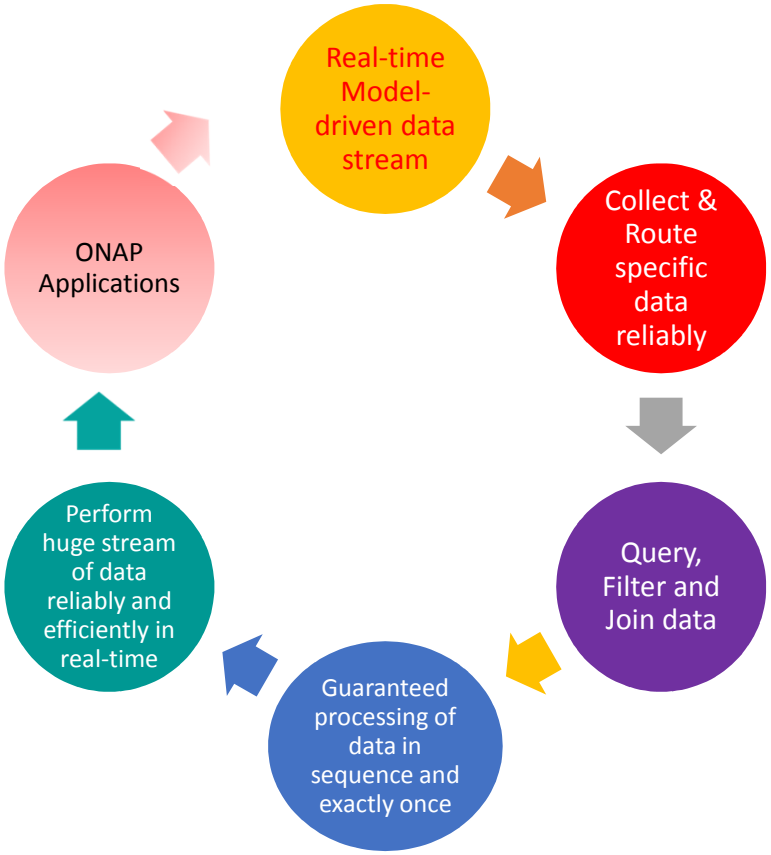
Runtime

Run -Time Deployment

1. DCAE Controller pushes instantiations configurations and policies for new services as defined in the new DACE Orchestration (Cloudify) Blueprints
2. Run-Time VES Real-Time Event processing by use cases Microservices

ONAP Gaps (1/2)

Real-time Data Streaming and Data Processing Pipeline



Rich Complementing Common Service Beyond a BUS

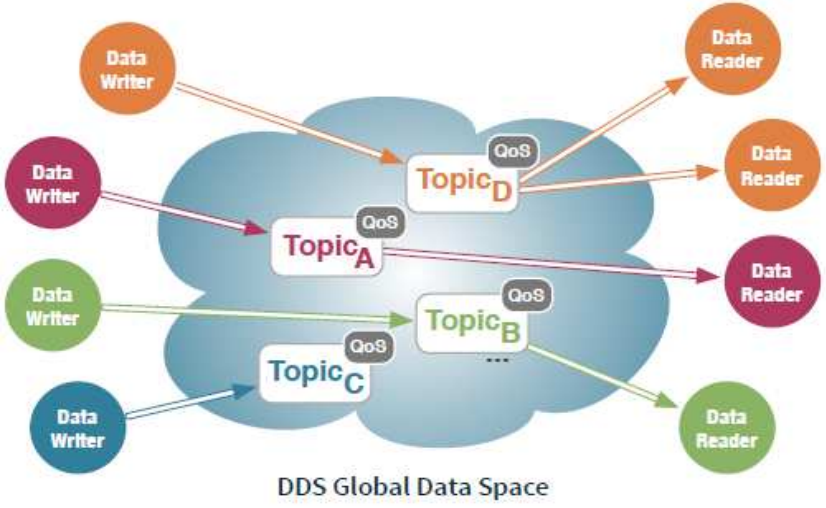
1. Real-time Data Model based event Streaming for north to south
2. Collection real-time data event stream and routing of stream for actionable processing
3. SQL operations capability on the real-time data stream
4. Exact once streaming of real-time event data for sequential processing
5. Fire hose event data streams processing in real-time and reliably
6. ONAP applications enablement through real-time library-based pub/sub

ONAP Gaps (2/2)

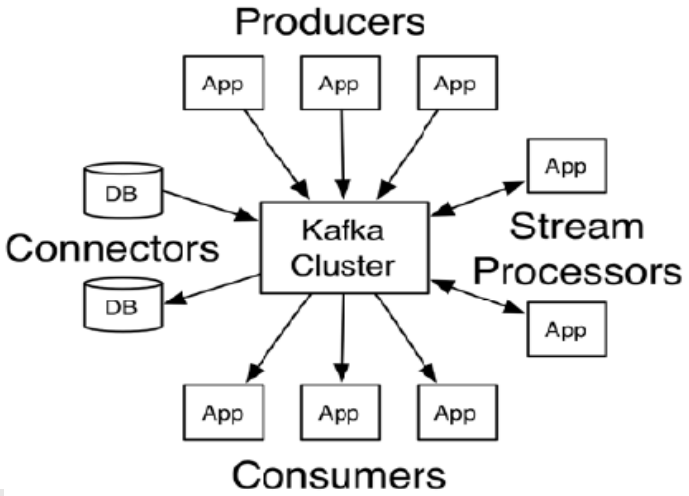
MSB BUS	DMaaP /Kafka	DDS Real Time Streaming
Microservices Operations API centric	N/A	N/A
JMS capability to handle microservices health operations	Pub/Sub **Message** oriented middleware, JSON based payload, centralized	Provides a rich Model driven binary **Data centric** Pub/Sub Geo **Distributed** Domain based capability to provide full microservices interconnectivity beyond Java
No Native Data Synchronization	No Native Data synchronization	**Logical data Objects**, changes automatically track to make the system track states and **self synchronizing**
Client/server	TCP	UDP/TCP with a **QoS** based wire protocol for optimization and reliability fine grain flow control, ***for very fast binary data type delivery and tracking of events***
		Concurrent, non-blocking, back-pressure handling end to end real-time pipelined system

Comparison: DDS vs Kafka

- Geo Distributed Global Data Centricity and Synchronization
- Low latency Pub/Sub P2P, Broker(optional) and N2M
- Push to Subscribers
- Query and Filter on data-value
- Data Centric
- Best effort, last-n, reliable and exactly-once
- 20+ QoS settings



- Broker Based
- Pull from Cluster
- Opaque Data
- Message Centric
- At least once, at most once, exactly once
- Controlled reliability and durability



RESP Interfaces and API calls (1/2)

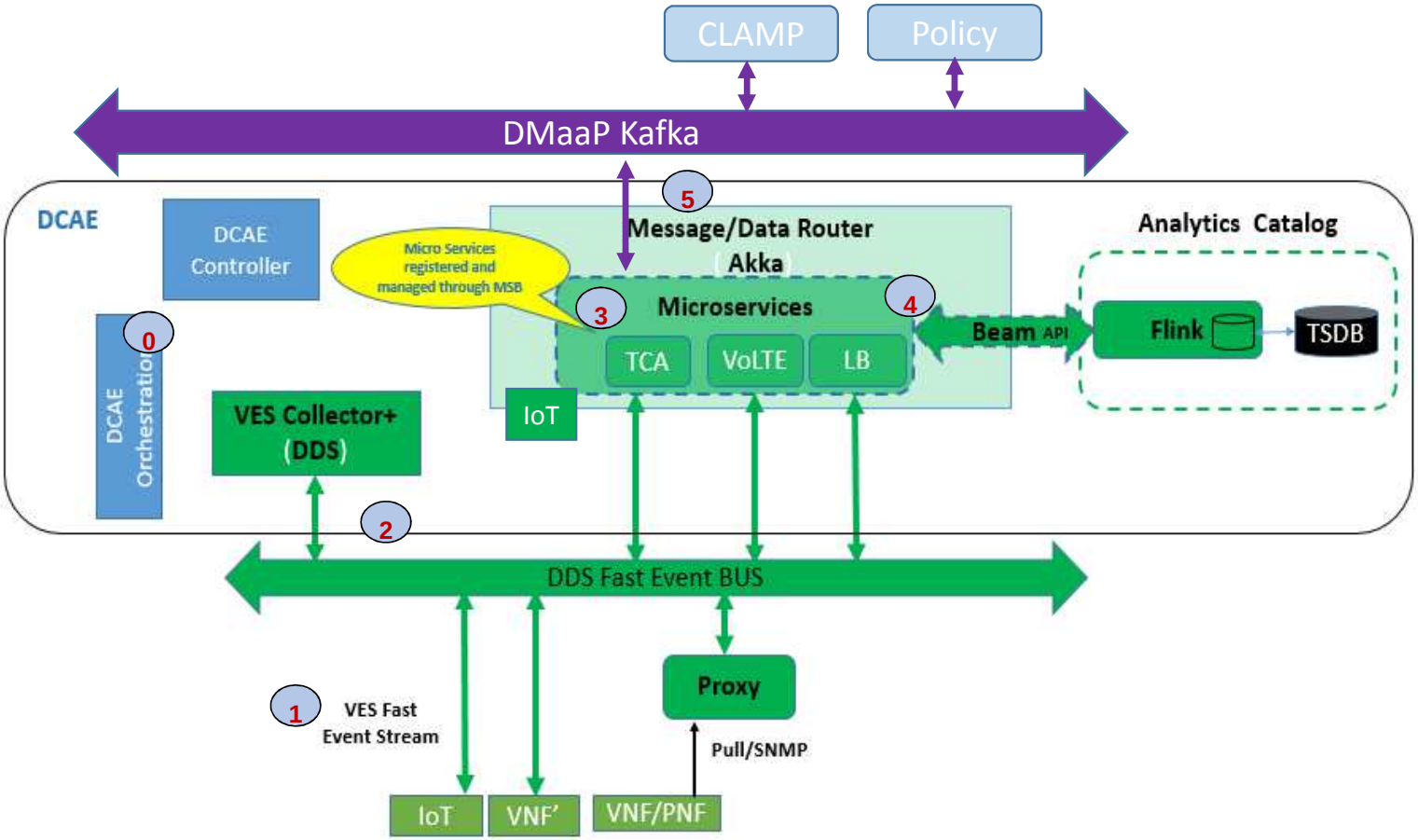
- **Configuration & Operations**
  - Artifact design and creation mapping to model-driven network services/VNFs for pub/sub topics (e.g., VES and other ingestion sources)
  - A&AI – resource inventory
  - Service and resource orchestration
  - Ongoing real-time updates for new services and/or changes to existing services
- **Run-time Operations**
  - Real-time event streams based on pub/sub configurations from VNF
  - Main application Akka actor-based concurrent handling of streaming event with DDS embedded library real-time event streams, and cache DB store
  - HA and error handling
  - Real-time Beam SDK calls to Flink real-time analytics tools

Interfaces

- **API calls to push configuration information to the RESP controller**
  - Pub/Sub topic push
    - Analytics
    - Filter, Search, Group by resources
    - Rules for actions for component interactions
- **Input run-time real-time event streams**
- **Output real-time event streams**
- **API calls (if needed)**

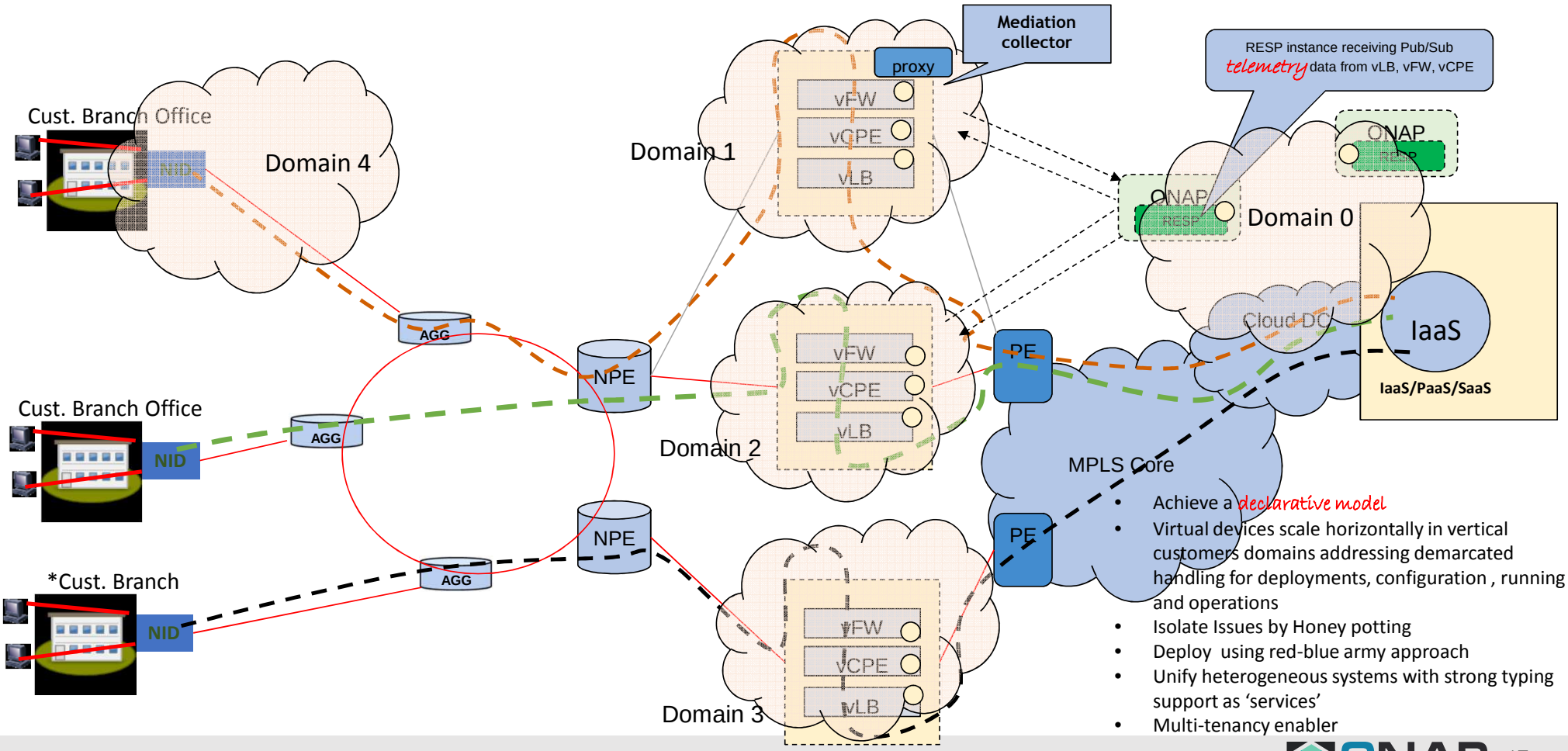
RESP Interfaces and API calls (2/2)

0. DCAE Control components
   - Driven through new TOSCA blueprints by Cloudify/Orchestration
1. DDS Enabled with VES Event Streaming Topics
2. VES Collector
3. RESP Service Components
   - Use Case Microservices
4. RESP Platform Components
   - Flink with Beam API
5. RESP Microservices interact with DMaaP/Kafka



DDS Domain-based Network Slicing Example

Enterprise Use Case



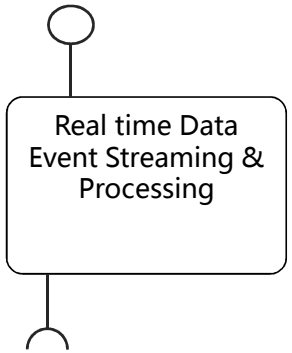
- Achieve a *declarative model*
- Virtual devices scale horizontally in vertical customers domains addressing demarcated handling for deployments, configuration , running and operations
- Isolate Issues by Honey potting
- Deploy using red-blue army approach
- Unify heterogeneous systems with strong typing support as 'services'
- Multi-tenancy enabler

Thank You

Backup Slides

RESP Functional Description for Casablanca

- Data Collection interface
- Deployment Interface
- Config binding interface



- VNF/Proxy Fast Data event stream
- Data enrichment interface (A&AI)
- Service model change interface (SDC)

Definition:
RESP is ONAP/DCAE subsystem that enables Fast Data Event Streaming, Fast Data Event Stream based higher level analytics and correlation for business and operations activities. RESP functionality will allow for fast data based collection of performance, usage and configuration data; provides handling of fire-hose event source in real time along with analytical processing for supporting operational decisions, trouble shooting and management; provides fast track results through pub/sub data event stream publications to rest of the ONAP system for FCAPs and other functionality

Provided Interfaces:
- Interface 1: Data collection interface (provided by DCAE collectors, consumed by VNFs and others)
  - Interface for various FCAPS data entering DCAE/ONAP.
- Interface 2: Deployment interface (provided by DCAE Deployment Handler, used by CLAMP and other northbound applications/services)
  - Interface for triggering the deployment and changes of a control loop
- Interface 3: Configuration Binding Service
  - Interface for querying the information of the services that are registered to DCAE Consul
- -Interface 4: Data collection interface (provided by RESP/DDS collectors VES, Proxy and others)
  - Interface for various FCAPs, IoT other data entering DCAE/ONAP.
- Interface 5: Deployment interface (provided by DCAE/RESP Deployment Handler, used by IoT and other northbound applications/services)
  - Interface for triggering the deployment and changes of a control loops, IoT TCA and other use cases
- Interface 6: Configuration Binding Service
  - Interface for querying the information of the Fast Event based services that are registered to DCAE/RESP Consul

DCAE interfaces leveraged & extended

Consumed Interfaces:
- Interface 1: Data movement platform interface (provided by DMaaP)
  - Interface for data transportation between DCAE subcomponents and between DCAE and other ONAP components
  - This interface can also be used for publishing events to other ONAP components.
- Interface 2: Data enrichment interface (provided by A&AI)
  - Interface used by DCAE collectors and analytics for querying A&AI for VNF information for the purpose of enriching collected raw data by adding information not contained in original data.
- Interface 3: Service model change interface (Provided by SDC)
  - Interface for DCAE/RESP Service Change Hander fetching control loop models and model updates for RESP apparatus (Akka enabled microservices, BEAM SDK, Flink, TSDB)

DCAE interfaces leveraged & extended

Consumed Models: TOSCA models descripting IoT & control loop construction (e.g. collection and analytics apparatus)

DDS for Geo Distributed Data Synchronization (DB)

- Figure 1 depicts cross DC deployment
  - Scenarios 1-4 reflect the network link state
  - The data flow across network tracked reliably for both Deterministic and Non-Deterministic network links
  - Geo Data synchronization guaranteed through Eventual Consistency for apps as shown in Figure to provide Geo distributed Event store
  - Light wire protocol allows for low latency times in the order of < 1msec over connectionless interconnect and with binary strongly typed data with QoS for reliability
- Figure 2 depicts a DDS based application deployment
  - Shows efficient query & filtering capability
  - Event DB store as Cache/persistent for logical objects
  - Very Rich meta-data information on each event
  - Each App has an embedded library

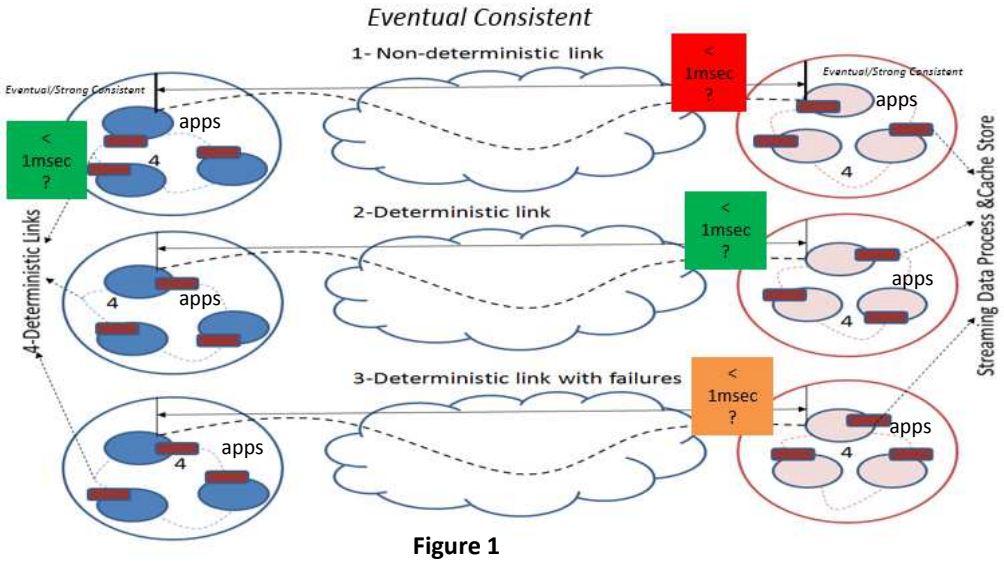


Figure 1

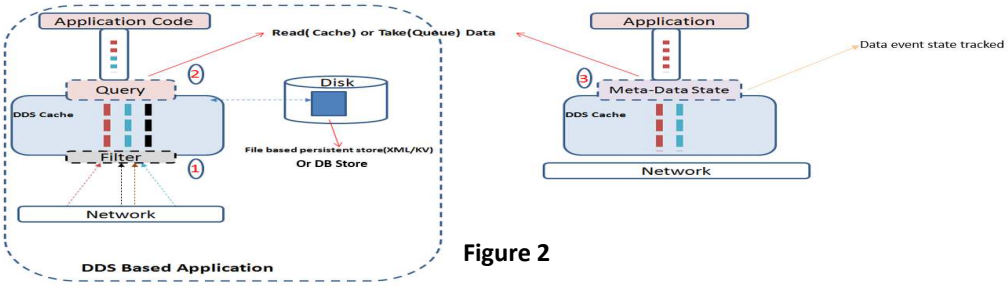
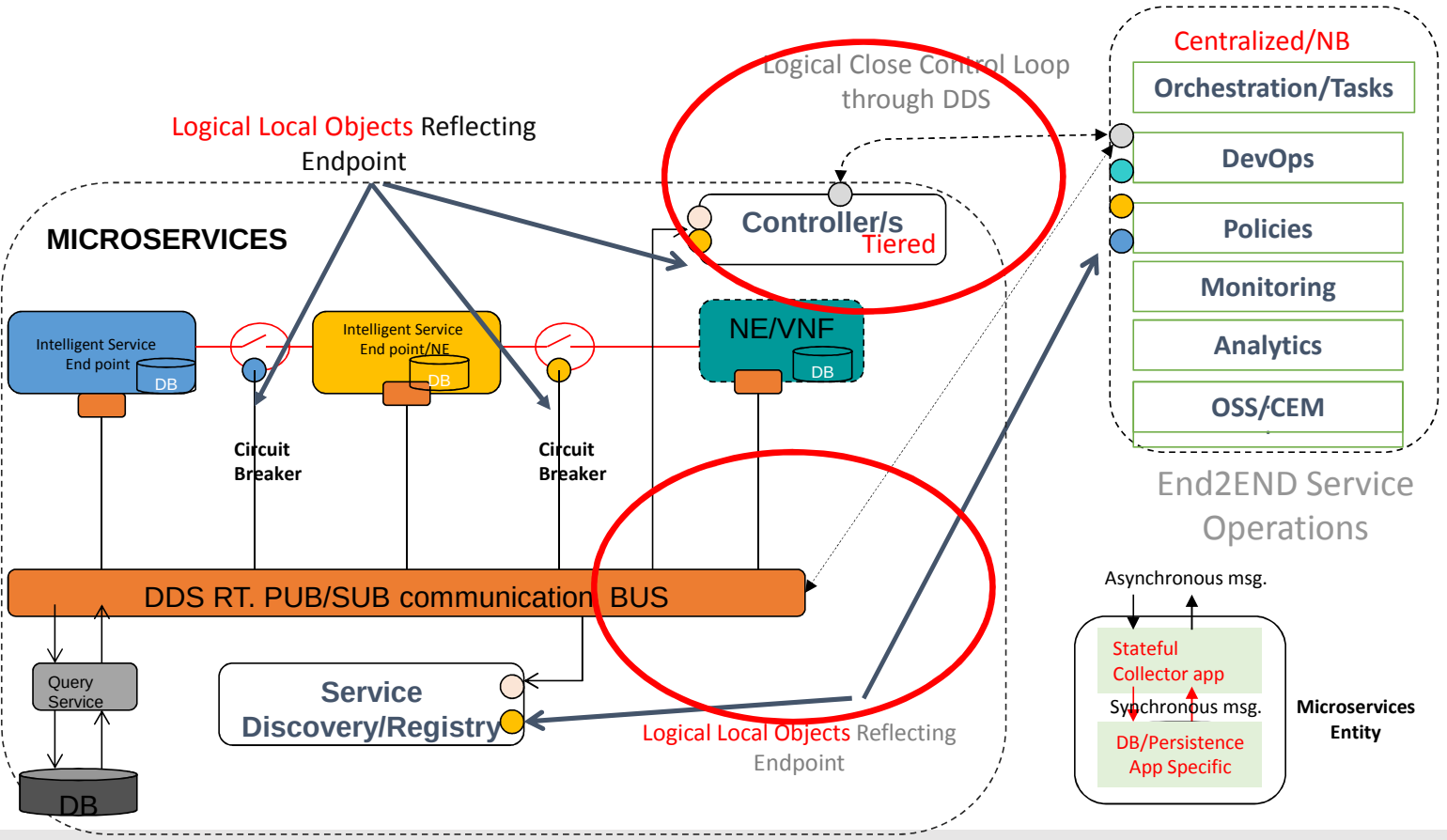


Figure 2

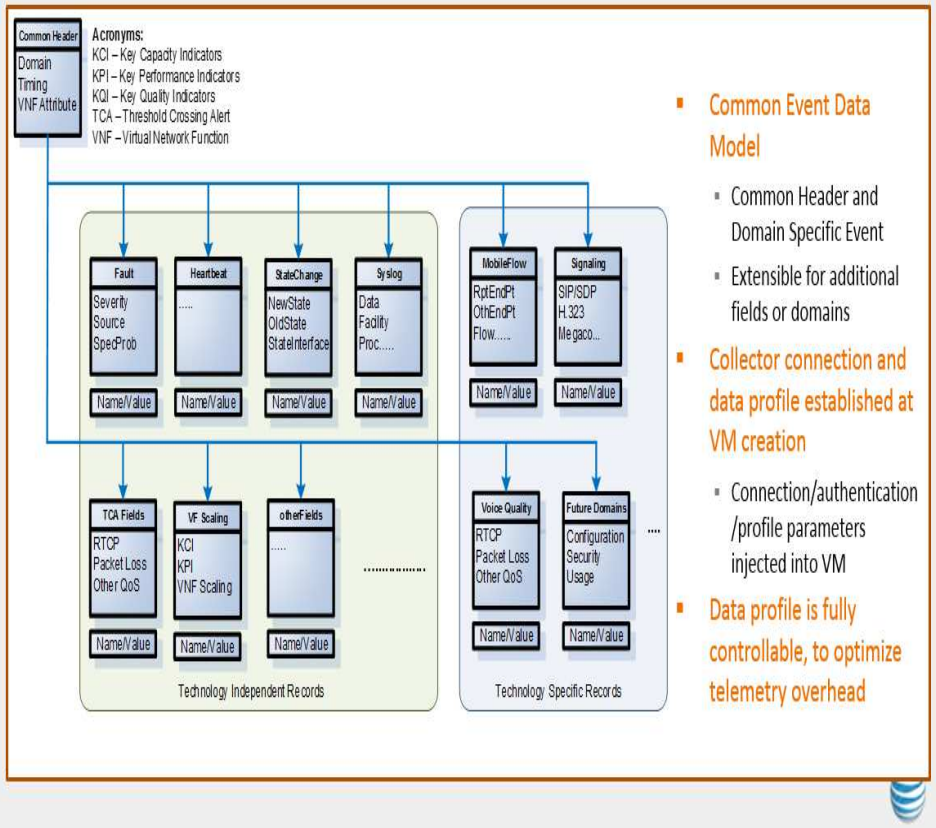
Unifying Fabric – Microservices Centralized and Edge Interconnect
(Logical Object based State Synchronization)



- MSB BUS Internal and External Event based BUS
- DDS fully supports *intelligent Interconnect* needs for IoT and SDN, PNFs, VNFs, a unifying fabric
- Logical *local* objects at the Publishers/Subscribers dynamically reflect/ track Endpoint Telemetry/State/Other in real-time asynchronously
- Centralized NB systems, Controller/s, NE/VNFs, Service Discovery, E2E Operations all connected through DDS Data Domains
- A Key DDS based interconnects can be sharded and tiered per Pub/Sub to provide a microservices loosely coupled env.

VES Mapping

VES – Common Event Data Model



VES Requirements Mapping
✓ Overall common Event Data model, Event Stream and collection architecture
✓ OPNFV support for the VES Common Event Data Model
✓ Consisting of a Header and an Event Specific Block, with additional Name/Value fields for extensibility
✓ A VNF Event Stream (VES) Common Event Data Model
✓ A VES Agent that can collect the VNF Event Stream data from the VNF and deliver it to the VES Collector
✓ A VES Collector that can consume the VNF Event Stream data, and store the data in a database backend
✓ *VES plugins for integration with OpenStack infrastructure services such as Monasca and Vitrage (*
✓ Common Event Data Model:(Fault, Custom, programmable), *authentication*, access

DDS for Common Data Bus	RTDESP
Common Data model with declarative IDL, Extensible types P2P Pub/Sub	Common Event Stream, and beyond
Same API exposed for all HW, OS Language bindings (Java, C++,C,C#)	Reactive real-time Processing and Batch Processing
Dynamic Discovery across Domains & Topics accommodating common header with profile creation through IDL , fulll control	Distributed
Rich QoS policies, allow for fine grain communication control beyond Transport(very low latency support, jitter support) . Many knobs for fine tuning	Concurrent , HA , Secure
Strong Type support, , application read(), write() calls with specific data types, can support different payload with defined types (different protocols as payload)	Scalable end to end processing for edge, Lake
Open standard and proven Interoperability	Many to Many (tiered model), shards, partitioning, Domain based
Query support for Real time filtering of Events, Complex Event Processing, UDP, TCP	Persistence/Store (Cache, DB)
Secure (SSL/TLS)	Integrated collection and integrations beyond Telemetry for heterogeneous networks
Auto Data Endpoint changes discovered, with high Priority (QoS)	Flexible extensible framework through DDS enhance Agents and VNFs

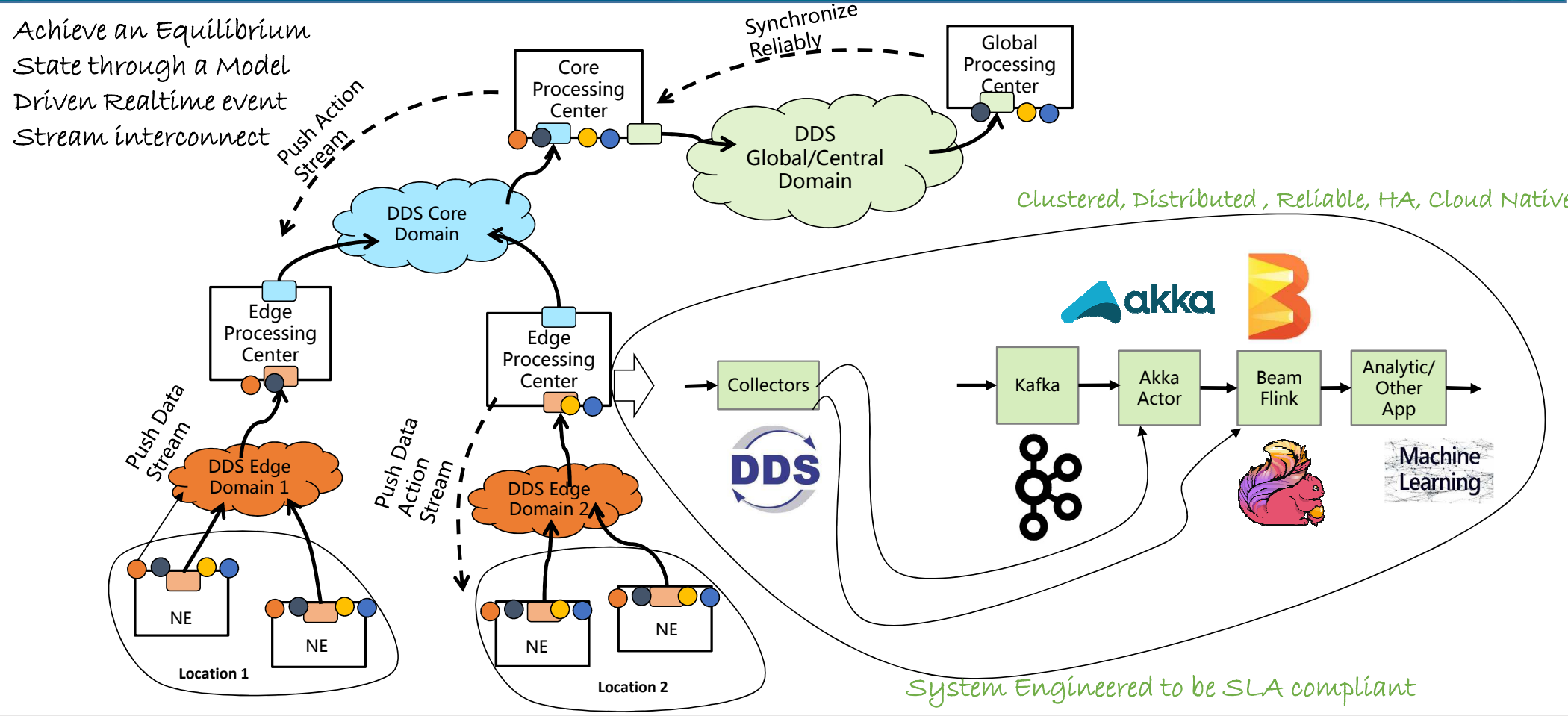
ONAP Gaps *for* Distributed Streaming and Data Synchronization

Core features	DDS	Kafka	Kafka Streaming	
In Memory Distributed DB	✓ Data/Object Centric DB Store, with Data Query Language support	Log based, Message Store, no visibility at the Data level	Query layer based on Message Store, no visibility at the Data level	Further DDS provides "Eventual Consistency" based data synchronization in Global Space providing consistent *state*
Integrated Efficient secure publish-subscribe binary-level communication	✓ Asynchronous Pub/Sub with distributed Domain based partitioning, data event Pub/Sub is strongly typed(binary) and state oriented	Text-based events, JSON based encoding , overhead communication and processing	Text-based events, JSON based encoding , overhead communication and processing	Further DDS DB cache and pub/sub are intelligent, as DDS tracks state of the data events stored. It allows for application to become model driven through well structure data models
System event reporting and logging	✓ Data/Object Centric DB Store, with Data Query Language support Rich Pub/Sub event tracking, reporting and logging with fine tuning through knobs for optimizing the data event traffic	Relies on TCP for flow control	Relies on TCP for flow control	DDS provides rich event tracking metrics for optimal data event traffic in the network on both Publisher and Subscriber end

ONAP Gaps *for* Distributed Streaming and Data Synchronization

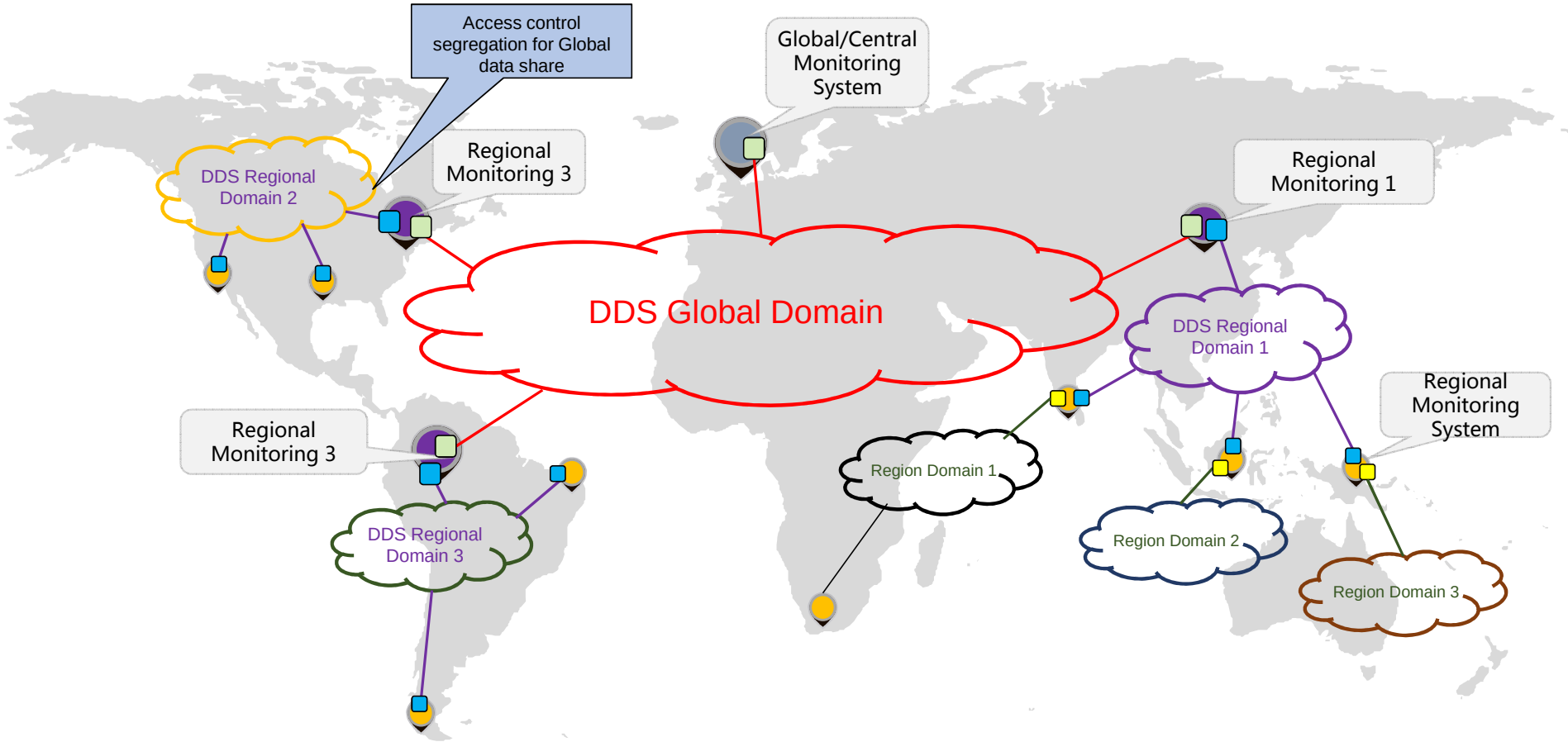
Core Features	DDS	Kafka	Kafka Streaming	
Distributed authentication and access control	✓ Distributed Authentication and Access control	✓ Authentication	✓ Authentication	DDS allows fine grain access and control Domain->Partition-> QoS/Topic-> Subs/Pubs
Distributed scheduling and Processing management	✓ Process management is present, DDS tracks the Publishers and Subscribers and re-syncs data upon recovery/re-start	Not present	Not present	**DDS auto scheduling and re-start is an overlay, not present natively

End to End Geo-Distributed System (RESP)

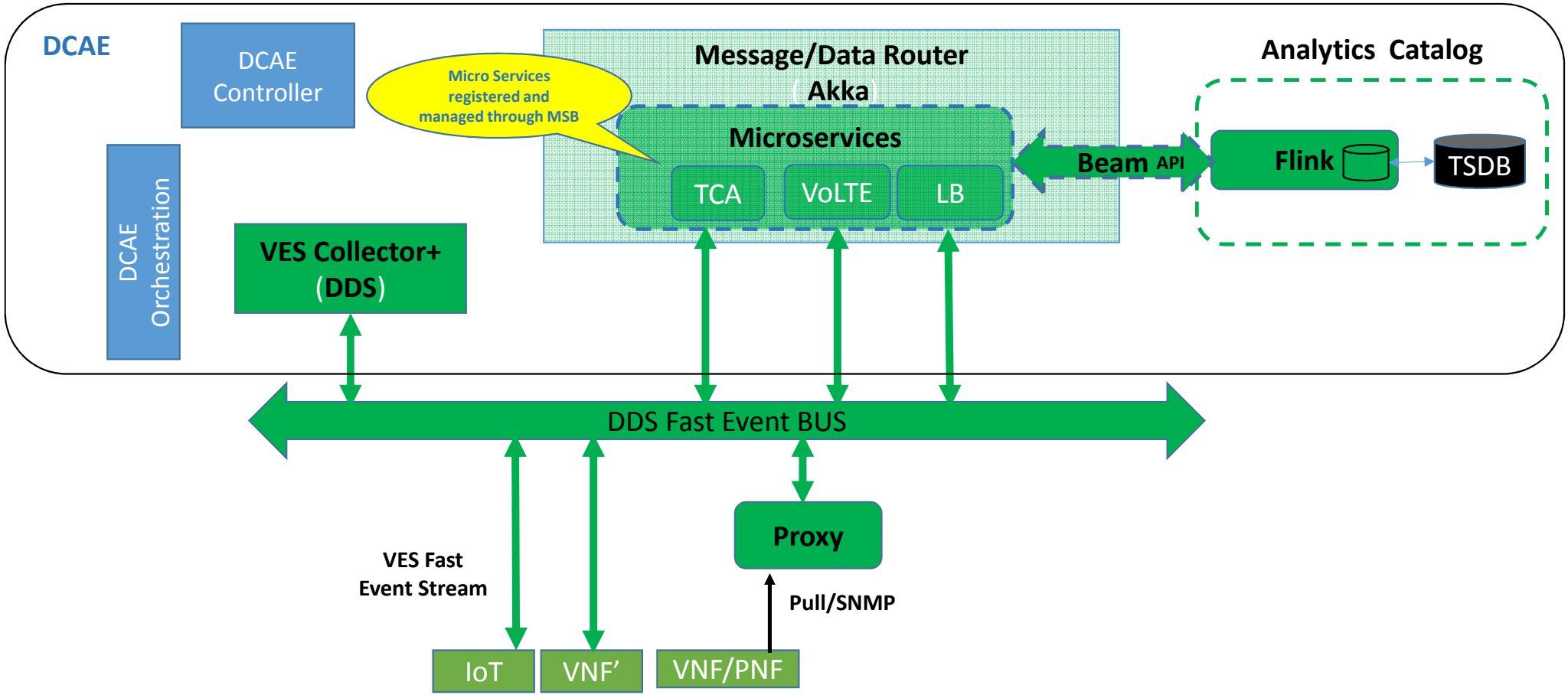


Geo-Distributed Tiered Deployment

(inter-Carrier use case)



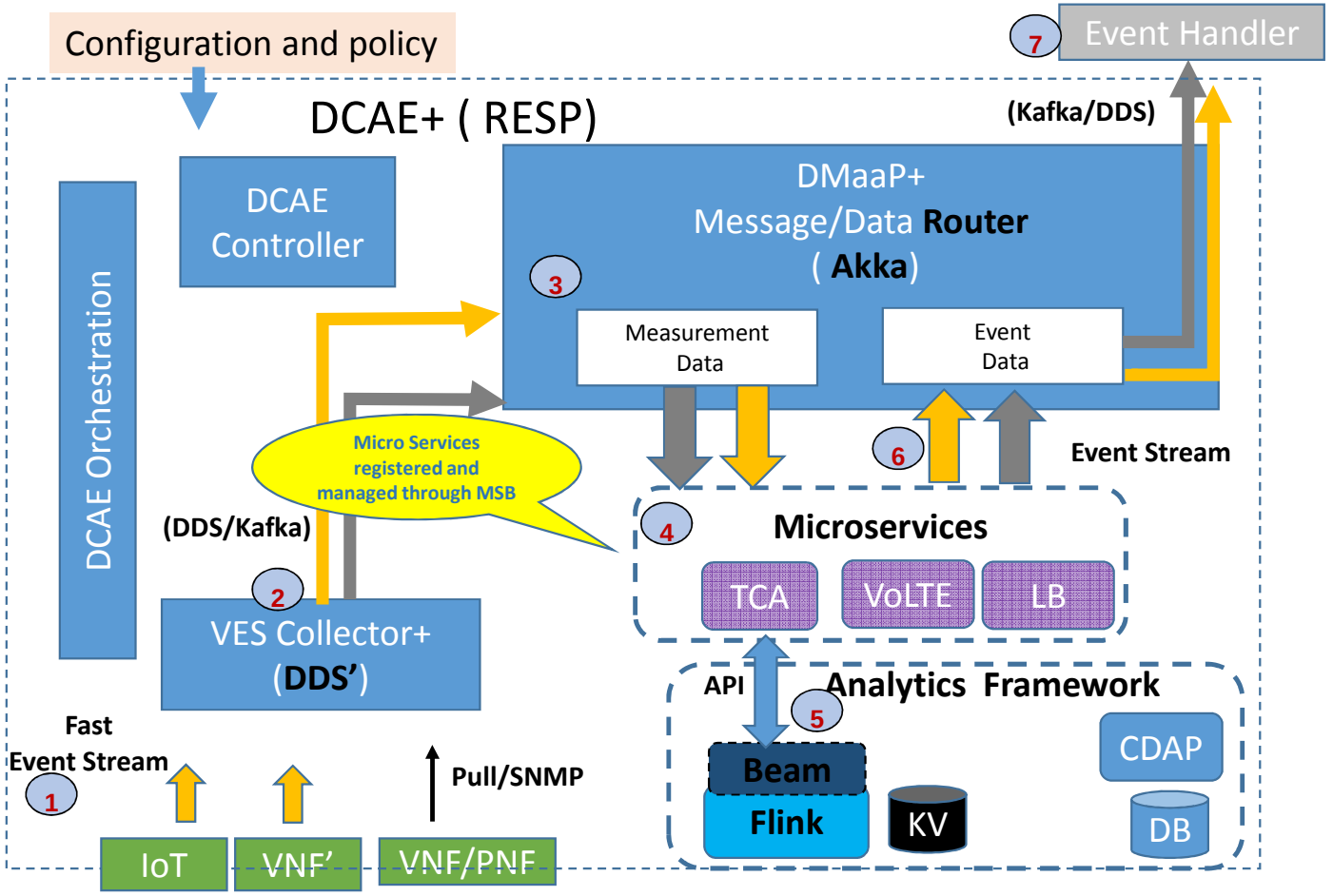
DCAE & RESP Integration



DCAE & RESP Integration

1. Data Streamed from southbound devices
2. VES **collector receives** RT stream & existing pull based measured data (acts as a proxy)
3. RT Stream can be directly received by DMaaP with **Akka** based routing
4. Microservices receive events and perform analytics processing using Flink through Beam APIs
5. Flink processing(Beam)
6. Microservices return processed result out of DCAE
7. As DMaaP Kafka/DDS event for Policy etc

Note: both existing and new DDS flows co-exist



ONAP RESP Logical Interface Flow

- RESP Logical Software building block tools
  - DDS, Akka, Beam & Flink
- Figure shows a flow sequence for RESP and some ONAP components
  a) A new end to end Realtime event streaming flow
  b) Current event flow can co-exist
  c) 2,3 internal RESP Routing and Processing
  d) 1,4,5,6 show external RT. and current event flow Close Loop
- VNF can push DDS based event stream
- Proxy can push DDS based event stream