An overlay architecture based on in-memory content delivery for funcX in edge-fog-cloud

Dante Domizzi Sánchez Gallegos

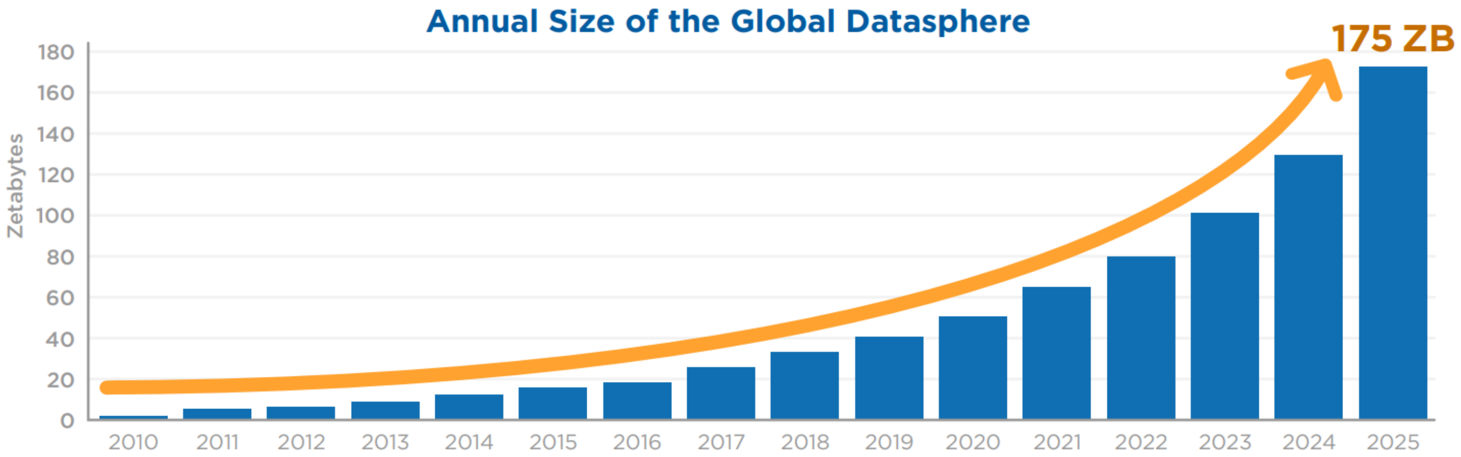
Cinvestav Tamaulipas

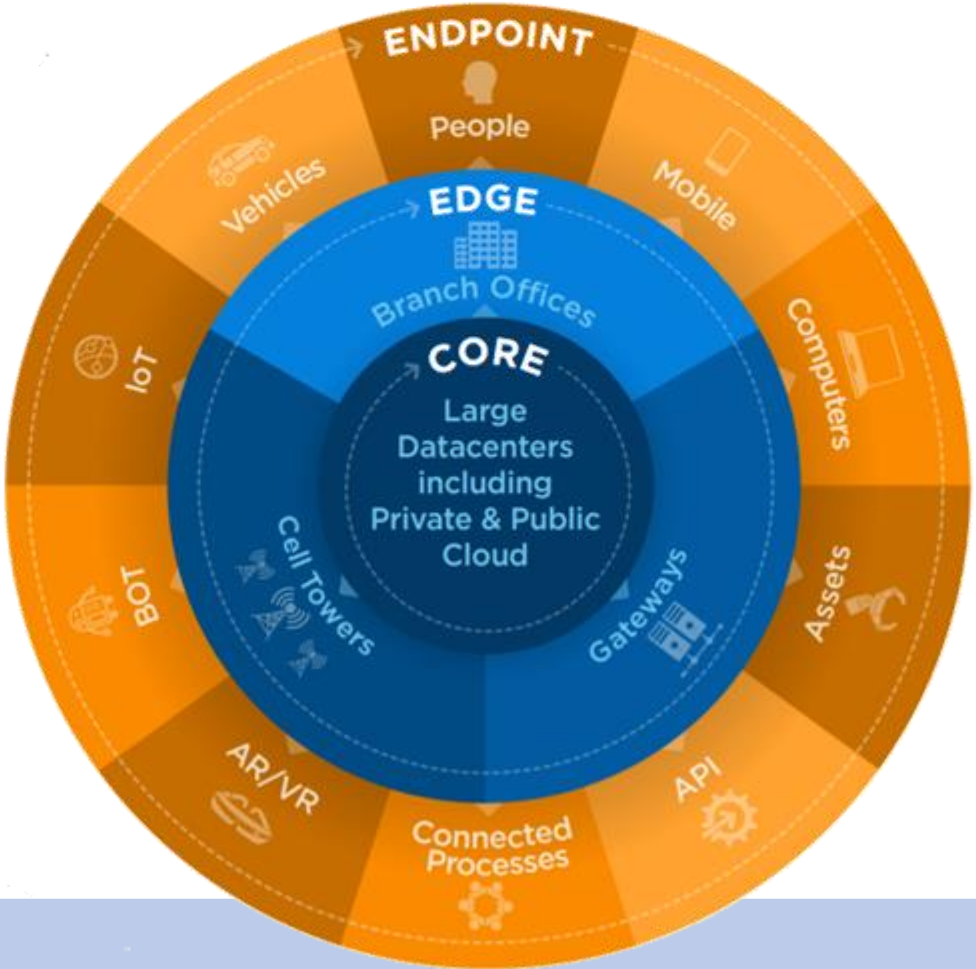
dante.sanchez@cinvestav.mx

Cinvestav
Tamaulipas

Motivation

The volume of the data is increasing in an exponential manner.

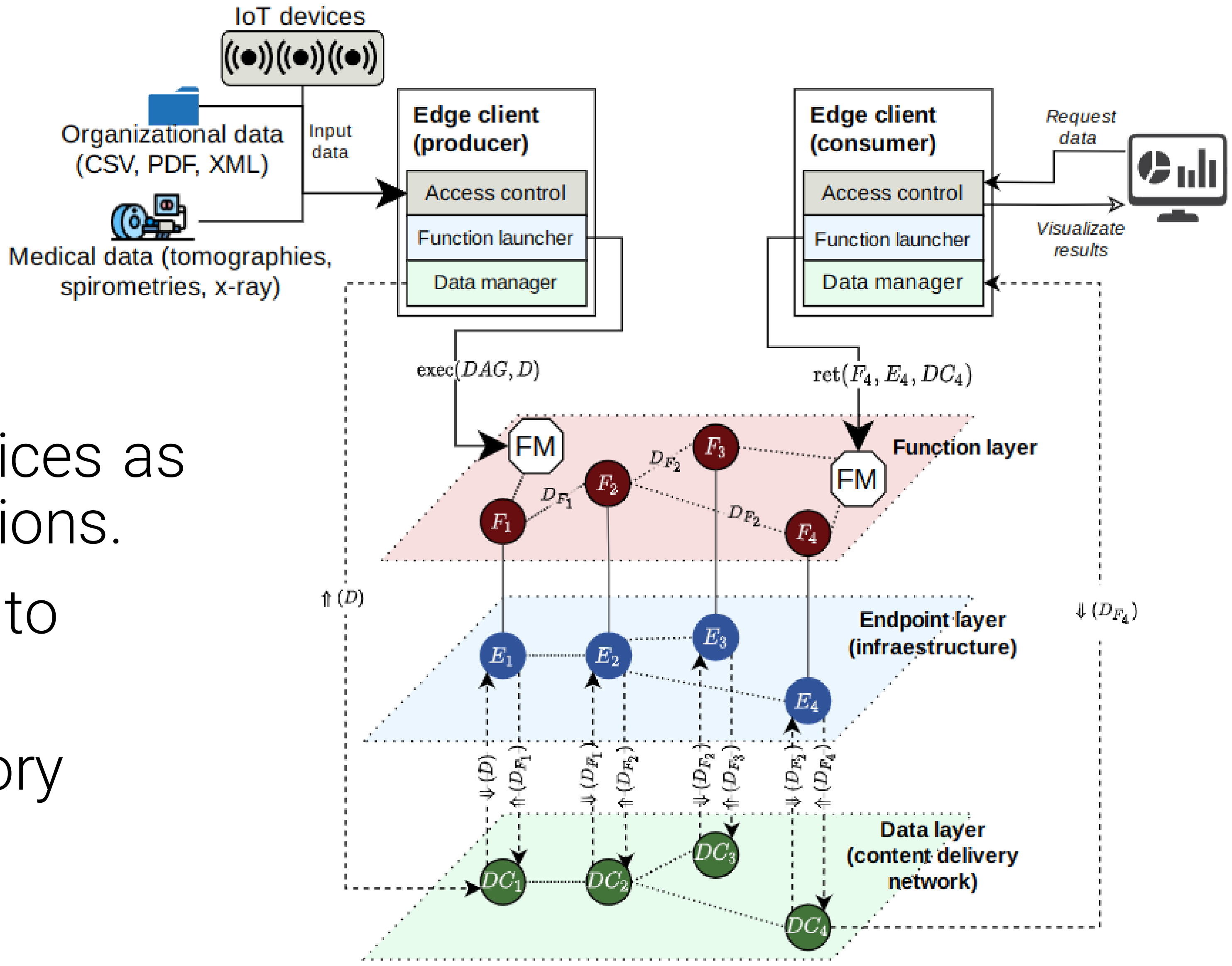




The data is managed and processed through different infrastructures (in any of the edge, the fog, or the cloud), rather than in a single environment.

Overlay architecture

- **Function layer**: design of services as a DAG by concatenating functions.
- **Endpoint layer**: infrastructure to deploy data and process data.
- **Data layer**: a CDN on in-memory storage.

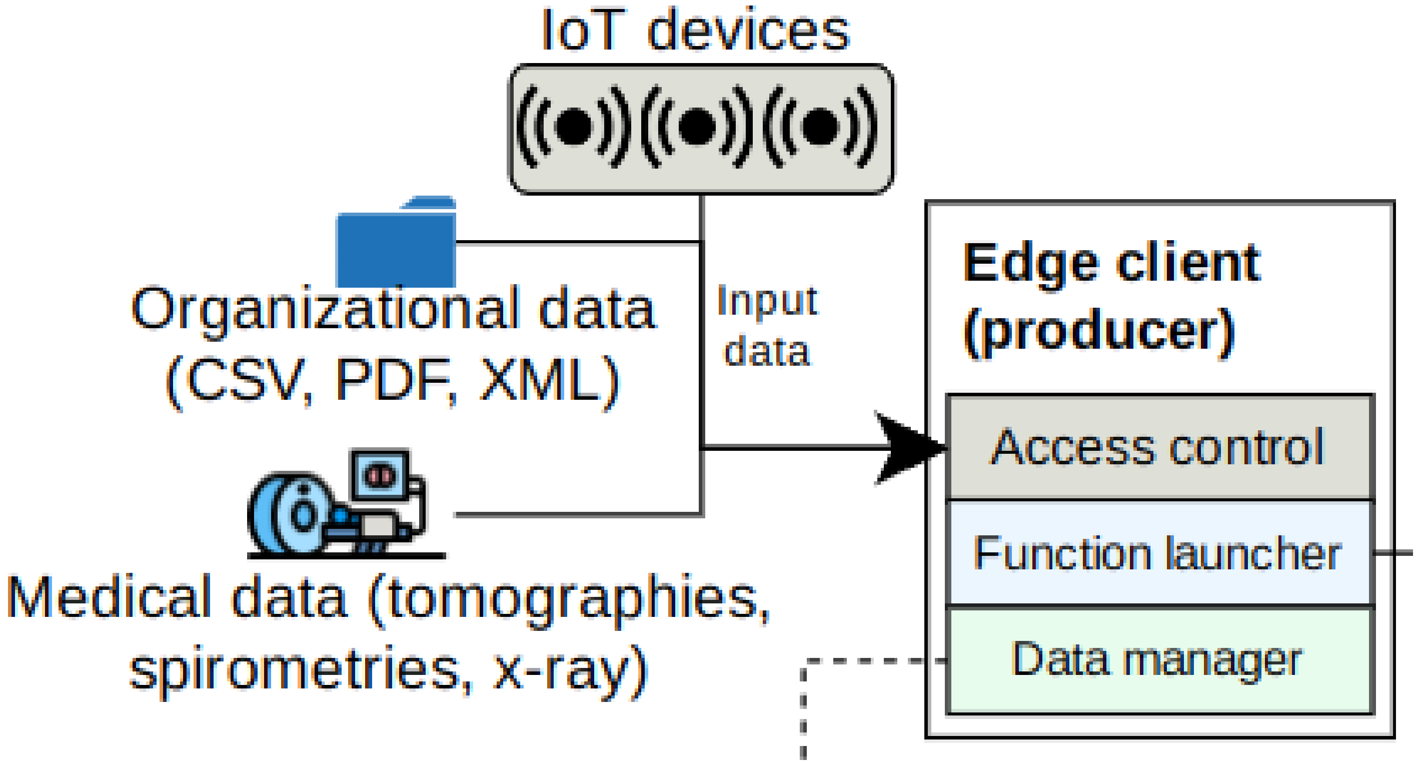


DC	Data container	FM	Function manager	ret	Retrieve operation
F	Function	$\Downarrow()$	Pull data	exec	Execute DAG
E	Endpoint	$\Uparrow()$	Push data		

$$DAG = [(F_1, E_1, DC_1) \rightarrow (F_2, E_2, DC_2)], [(F_2, E_2, DC_2) \rightarrow (F_3, E_3, DC_3)],$$
$$[(F_2, E_2, DC_2) \rightarrow (F_4, E_4, DC_4)]$$

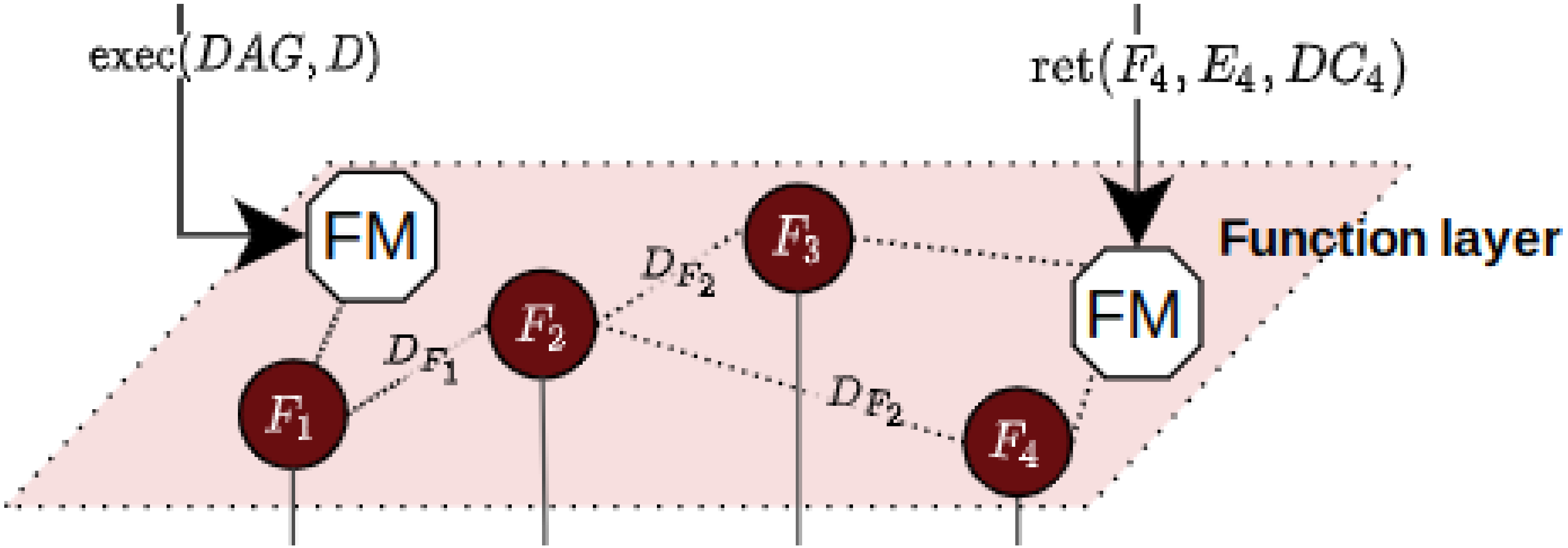
Overlay architecture

- The **edge client** implements a stack of services to manage the acquisition of data from different data sources (e.g., IoT devices, organizational data, or medical data).



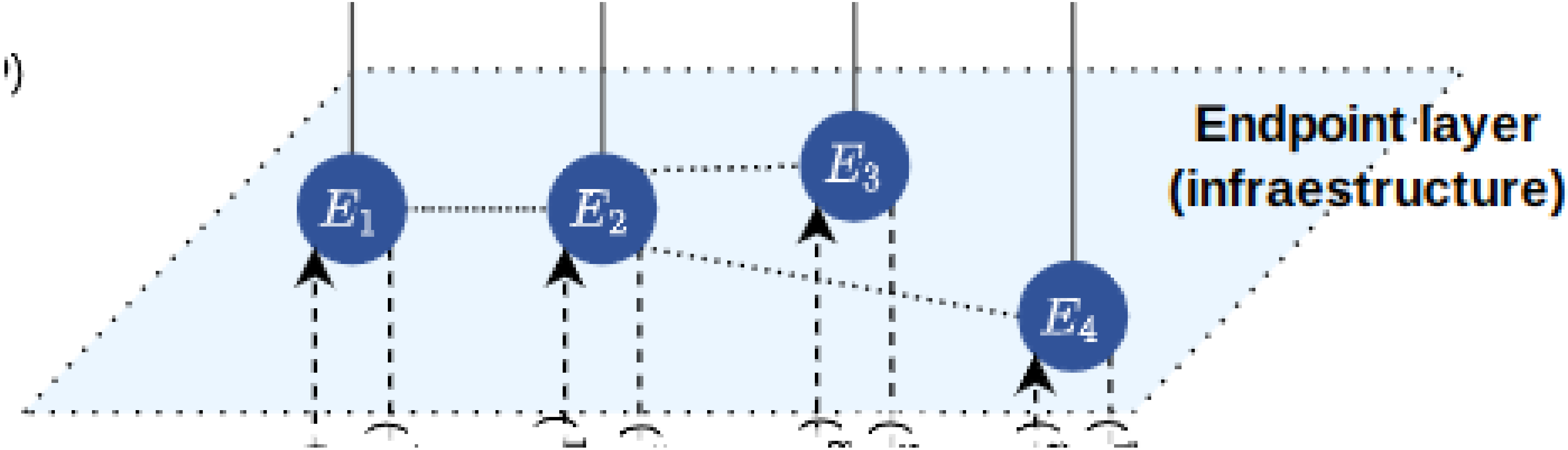
Overlay architecture

- The function layer is based on a FaaS mechanism, where users create and consume a set of functions to process their data and contents.



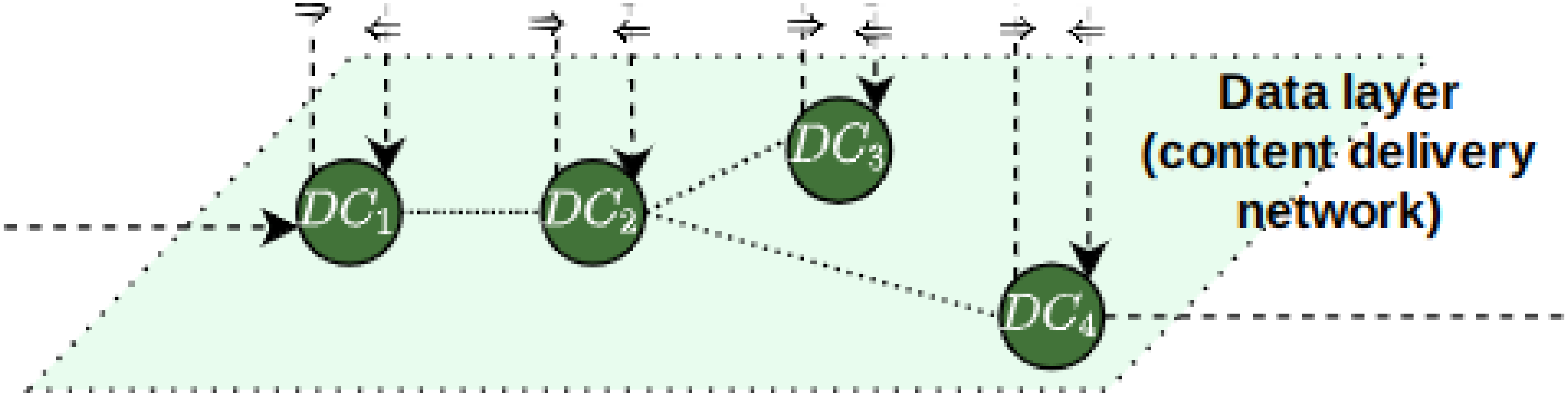
Overlay architecture

- This layer is implemented by using *funcX_endpoint* software, which performs the execution of the functions in the endpoint as well as the execution of the functions.

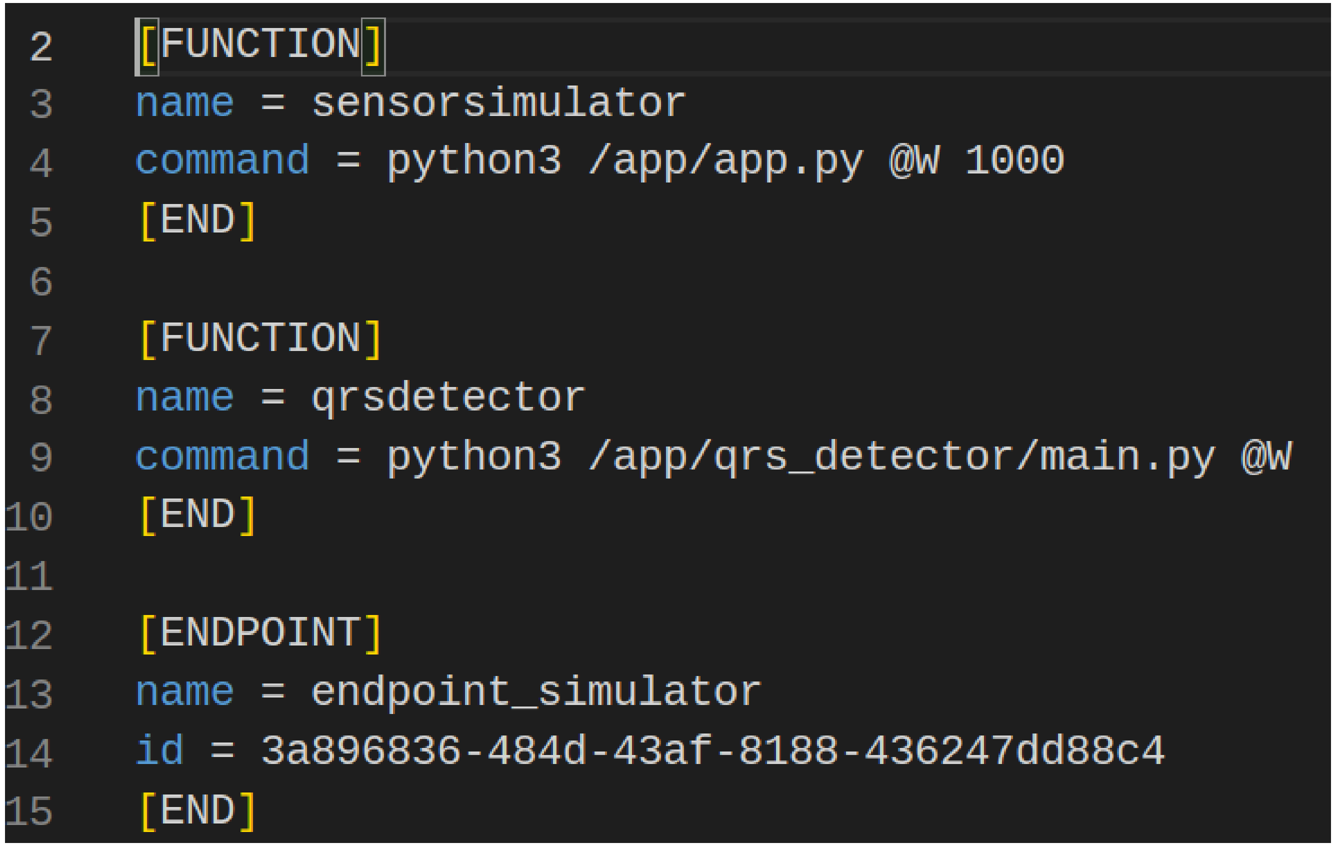


Overlay architecture

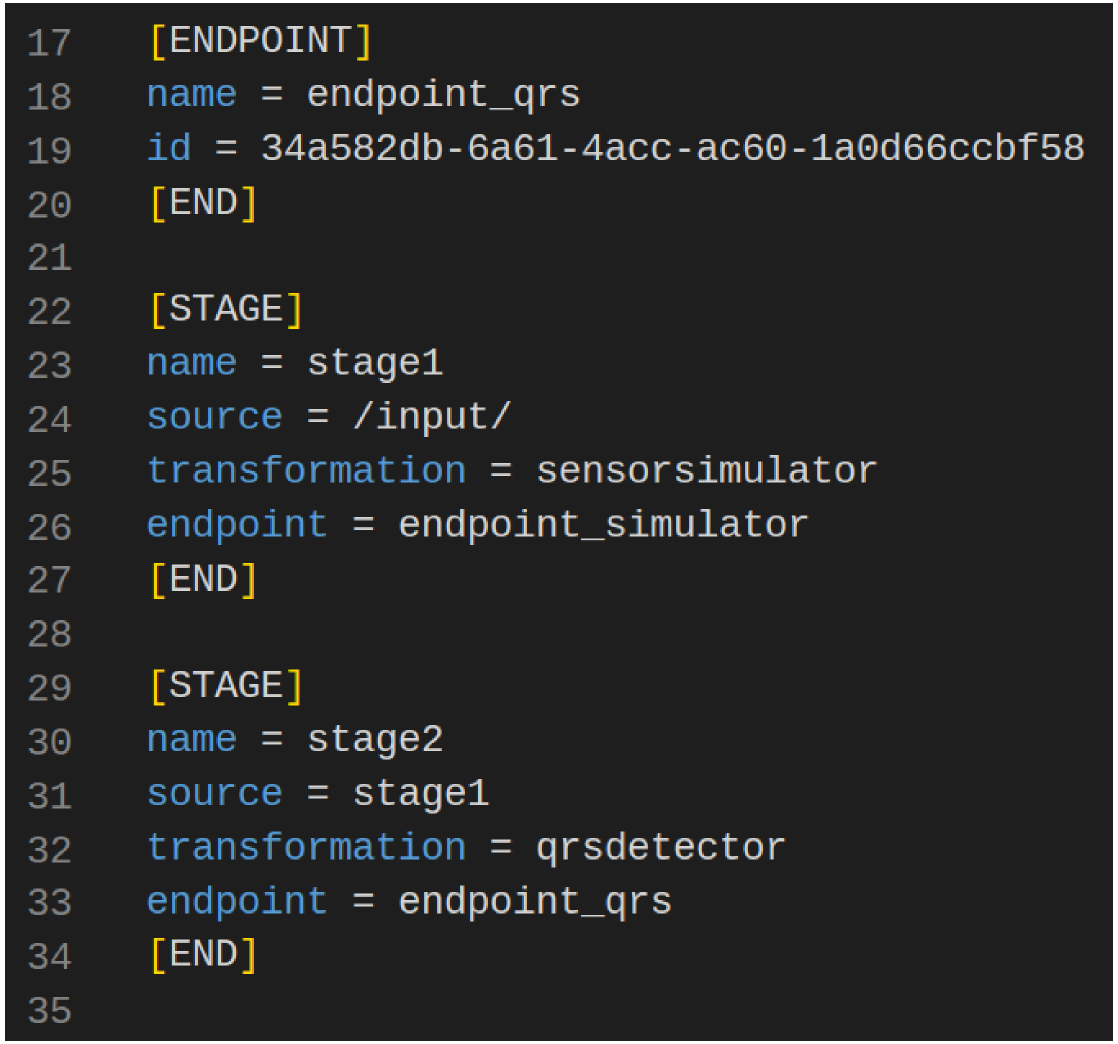
- This system is based on a pool of data containers that contains software structures that implement a temporal and hierarchical storage management.
  - First level: local memory (RAM).
  - Second level: local storage (filesystem).
  - Third level: cloud storage by using the CDN.



Declarative programming model



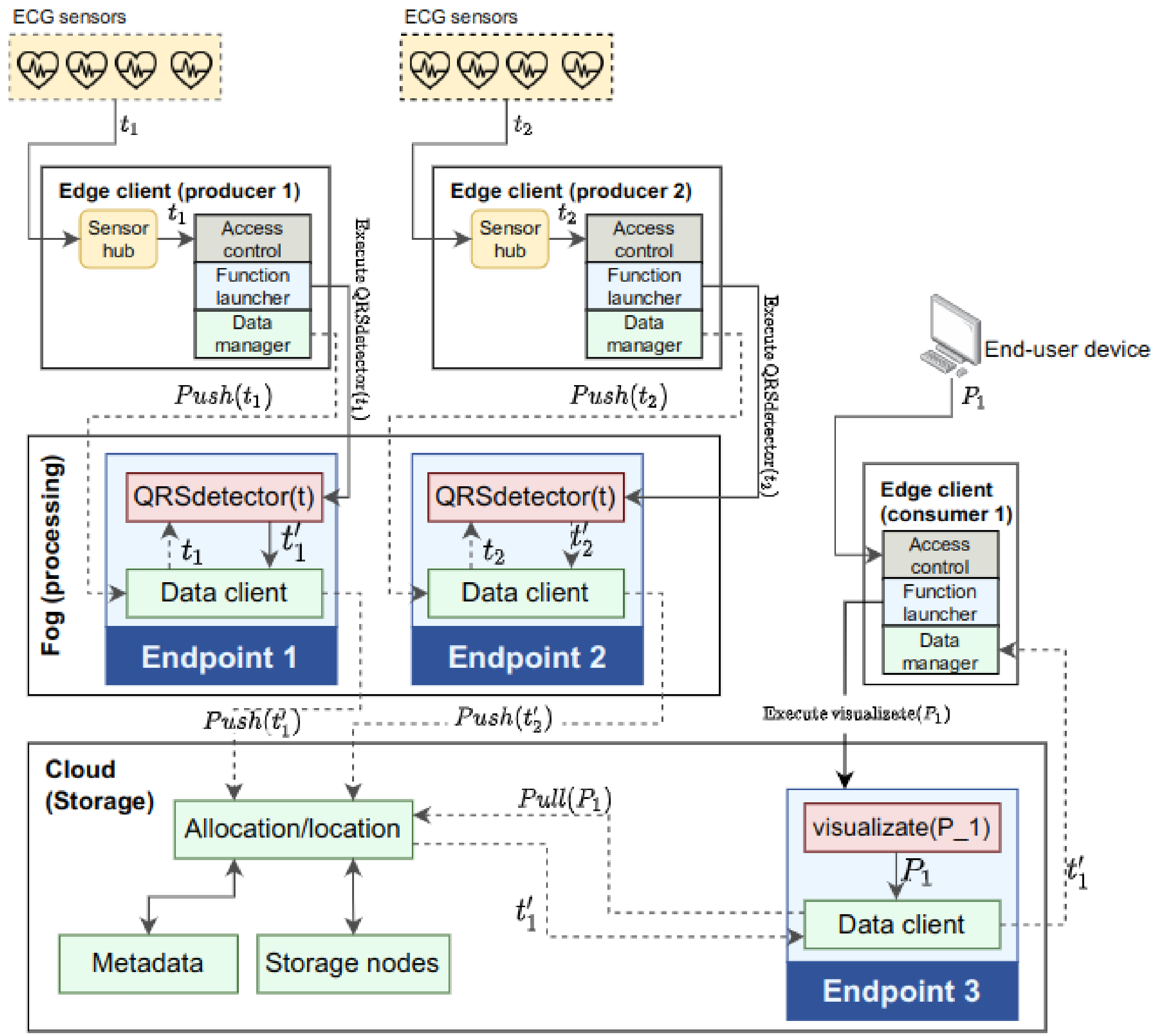
```
[FUNCTION]
name = sensorsimulator
command = python3 /app/app.py @W 1000
[END]

[FUNCTION]
name = qrsdetector
command = python3 /app/qrs_detector/main.py @W
[END]

[ENDPOINT]
name = endpoint_simulator
id = 3a896836-484d-43af-8188-436247dd88c4
[END]
```



```
[ENDPOINT]
name = endpoint_qrs
id = 34a582db-6a61-4acc-ac60-1a0d66ccbf58
[END]

[STAGE]
name = stage1
source = /input/
transformation = sensorsimulator
endpoint = endpoint_simulator
[END]

[STAGE]
name = stage2
source = stage1
transformation = qrsdetector
endpoint = endpoint_qrs
[END]

```

Use case of functions for the management, analysis, and storing of ECG data in the edge-fog-cloud

- The ECG data are collected by a **sensor hub** that store the data in form of text plain files.
- An edge client implements a funcX client, which invokes a function on a fog endpoint to identify QRS-complex.
- The results can be recovered by consumers (e.g., physicians or nurses) at the edge by invoking a visualization function



Thanks!

Questions?

Dante Domizzi Sánchez Gallegos
Cinvestav Tamaulipas
dante.sanchez@cinvestav.mx