Open Access

Open Journal of Internet of Things (OJIOT)
Volume 7, Issue 1, 2021
http://www.ronpub.com/ojiot
ISSN 2364-7108

Data-Centric Edge Federation: A Multi-Edge Architecture for Data Stream Processing of IoT Applications

Tiago C. S. Xavier, Paulo F. Pires, Flavia C. Delicato

Av. Gal. Milton Tavares de Souza, s/n, 24210-310, Computing Institute, Fluminense Federal University, Niterói, Brazil, {tiago.cariolano, fdelicato}@gmail.com, paulo.pires@ic.uff.br

Abstract

Emerging Internet of Things (IoT) applications demand data stream processing with low latency and high processing power. Although the cloud naturally provides huge processing capacity, high latency to move data to the datacenter is prohibitive. Edge computing is a recent paradigm where part of computing and storage resources are pushed from the cloud to the edge of the network. In edge computing, edge providers manage their resources near to IoT devices to meet low latency application requirements and reduce the network core bandwidth. To reach the maximum potential of edge computing, a big challenge is to promote the cooperation between edge providers. Currently, edge computing architectures are severely limited for providing cooperation mechanisms between distinct edge providers. In this paper, we propose a edge federation to leverage the cooperation between different edge providers. The edge federation uses interest information propagated in data streams that travel between edge providers to allow an stakeholder to react to inefficient resource allocation and service provision. The main objective of the federation is to create a consortium of edge providers to provide cooperation mechanisms and define and standardize the application interests. The proposed edge federation is (i) data-centric, since edge providers can share common interests and data and, thus, establish cooperation to increase the capacity to provide services for applications; (ii) distributed, since no assumption is made concerning the geo-location of the edge providers and their logical connections; (iii) opportunistic, because an edge provider can react dynamically to the environment change ; (iv) scalable, since the edge provider has the ability to analyze a data flow passing by its infrastructure and make decisions to increase network performance locally, which impacts the global performance

Type of Paper and Keywords

Visionary paper: *Edge Federation, Edge Computing, Resource Allocation, Archictecture, Cloud Computing*

1 Introduction

Internet of Things (IoT) is the paradigm that empowers physical devices to provide data collected from the real world to users, through the Internet, creating a dense integration of the virtual and the real worlds, where the communication between people and things takes place [9]. To meet the huge computational requirements of billions of IoT devices generating data, the cloud computing has been explored as the main backend platform for IoT since it can provide flexible and potentially unlimited storage and processing

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2021)* in conjunction with the VLDB 2021 conference in Copenhagen, Denmark. The proceedings of VLIoT@VLDB 2021 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

resources [3].

Data centers are the physical infrastructure where cloud data processing happens and are interconnected to cloud users by wide area networks. They are usually far away from end-users and not able to provide low latency and high bandwidth connectivity to emerging IoT applications [15]. Such applications are challenging traditional cloud architectures leading to the arising of the new edge computing paradigm.

Edge computing takes advantage of the computational capabilities distributed in the continuum that encompasses from data-producing devices to the cloud. In the edge computing paradigm, an edge provider is a stakeholder that keeps a small data center near to a data source with processing power to process application requests. Edge providers are usually private entities close to end-user and data sources, which provide services with low latency.

Being mostly private entities, edge providers have their own and conflicting commercial interests and establish private edge-computing environments to meet specific requirements of users from their own viewpoint [4]. An edge provider does have not the view of other edge providers' resources and its cooperation capacity is low.

The number of clients which an edge provider can serve is limited to the resources managed by it. To augment the service coverage, the edge provider needs to deploy a huge number of services, which requires acquiring and maintaining more computing capacity, i.e. edge nodes.

To execute resource-intensive and low latency applications, the edge provider needs great computing and communication capabilities. Such a stringent requirement increases the cost of processing data in the edge [16]. The cost to move and process data in the infrastructure of an edge provider also increases due to its typical low scale.

The building of an edge infrastructure in an independent way along with the lack of cooperation with other stakeholders cause unbalanced and under-utilized edge resources globally. A standalone edge provider has limited information about the global environment since it manages only its own set of edge nodes and resources. Thus, it cannot optimize or share the resources with other providers and the chance of collaboration among multiple stakeholders is limited [12].

To leverage the potential of edge computing, isolated edge providers should interact and cooperate with other edge providers. Cooperation allows to augment service coverage and distributed processing capacity, besides reducing the number of requests forwarded to the cloud. Therefore, there is a need of an approach based on the collaborative edge where edge providers connect to the edges of multiple stakeholders that are geographically distributed [12].

The current proposal consists of a interest-centric edge federation that aims to be an opportunistic, dynamic, distributed, open and data-centric solution for integrating edge providers and leverage the cooperation in the edge tier. The edge federation is a consortium where edge providers establish agreements and standardize the communication. The fundamental premise of our solution is that applications that generate data streams have interests in certain data. Based on this premise, we consider that edge providers can share common interests and data and, thus, establish cooperation to increase the capacity to provide better quality of service for applications.

In our proposal, the edge provider detects data stream flows passing by its infrastructure and uses the interest information associated with the application that generated the data stream. From the interest, the edge provider obtains information about the required services for processing the data stream. This information allows the edge provider to decide whether it is worth using its infrastructure resources to process such data stream. Our architecture is data-centric because the services are provisioned according the application interest, which is directly associated with the required data by application. The main key benefits of the proposed edge federation are to allow cooperation between multiple stakeholders and dynamic reaction to the interest of the data stream application. By promoting cooperation, workload balancing occurs between multiple edge providers and not only inside a unique edge provider. The dynamic reaction gives flexibility for edge providers changing quickly their resource allocation decisions and adapting to environment changes.

Edge providers that are not part of the consortium can join it at any time. When an edge provider does not wish to cooperate with another edge provider belonging to the Edge Federation, our solution allows them not to authorize cooperation.

The remainder of this paper is organized as follows. Section 2 presents challenges for Edge Federation. Section 3 discusses related work. Section 4 describes the system architecture. Section 5 presents the detailed architecture and their components . Section 6 shows the main operation of the system. Section 7 concludes this work and draws future research directions.

2 Challenges for Edge Federation Systems

The need for cooperation between multiple stakeholders leads to edge federation architectures, where different edge providers share physical resources and workloads in a peer-to-peer fashion [2]. In a multiple domain environment of federated edge nodes, the edge provider can forward computation tasks to nodes in external edge providers with more available resources, meeting service level agreement requirements of applications [2].

The cooperation between different edge providers is natural for edge computing since the premise of a centralized computing location in the cloud is not mandatory anymore. The cooperation based on edge federation enables the horizontal expansion of resources over dispersed locations [11]. However, in federated edge environments, the challenges for designing and deploying traditional edge computing systems increase. In following, we discuss the main challenges related to edge cooperation and data stream processing.

2.1 Edge Cooperation

Edge computing systems impose challenges that are exacerbated in edge cooperation environments. Some of the main challenges consist of service discovery for meeting IoT applications, partitioning and distribution of tasks for optimizing the resource utilization, and the mobility of IoT devices [10].

The challenge of service discovery occurs because not all edge nodes have all the services and algorithms available. Therefore, IoT devices need to discover, through some protocol, edge nodes that have required capabilities to perform a task or request. In the Edge Federation, this problem is exacerbated since edge providers must discover edge nodes of third-party able to provide the required service.

Partitioning and distributing computing models are also more difficult in edge cooperation environments. It is more laborious to distribute and process data in an edge node located in another domain since the node resources are weakly managed. In addition, there is the challenge of aggregating the data generated by the data analysis of the different stakeholders and returning the results to the application with the lowest possible latency. Both task distribution and result aggregation in the edge are tasks that become more critical when required by time-sensitive applications.

Regarding edge mobility, the main challenge is to consider that devices can enter the system through an edge provider and exit from the system by another edge provider. The federation should provide connectivity in a transparent way while the user keeps accessing the system.

2.2 Data Stream Processing

One of the most challenging problems in edge computing systems is the analysis of the large number, large-scale and high-speed data streams generated by the IoT devices. Three factors make processing data streams challenging on edge computing: rate variability, dynamism and data quality.

The first factor is related to data stream arrival, which occurs at different rates and varies over time. For example, the frequency of a GPS sensor can be updated over a period of seconds, while a temperature sensor can be updated every hour [8]. Even a single IoT device can operate at different rates, such as a vehicle traffic control camera, which can provide both periodic static images or continuous road environment video , depending on the user's requirement. The main problem with the variation in the rate of data streams is the risk of information loss [8]. If the system is not prepared to receive the data at the time it is generated, the data may be lost.

In an Edge Federation, an edge provider does not handle the infrastructure of another edge provider. In this scenario, the management of multiple data streams arriving from different domains, with high rate variability, demands precise synchronization between federation participants to avoid data loss.

The property of changing the data stream according to the place in the space or time that it was generated is related to edge computing dynamism and also a challenging issue for Edge Federation systems. In smart cities, for example, autonomous car data change depending on where the vehicle is, or the time when the data was collected. An architecture that promotes the edge providers' cooperation should decouple the data location from the data processing location for tackling the data dynamism problem. For example, data creation and processing can occur in multiple edge providers.

Finally, data quality is one of the most important factors for the data stream processing of IoT applications. As Karkouch et. al. [6] pointed, data quality is the suitability of data gathered from IoT devices to provide services to users. In federated edge computing, data quality is more difficult to achieve, as the various heterogeneous sources generate different data qualities which will be processed by different infrastructures.

3 Related Work

Authors in [14] proposed to stream video from central cloud to end users as a solution to increase the quality of experience (QoE). The authors' proposal is an Edge-Cloud federation, called Federated-Fog Delivery Network (F-FDN), which is composed of several Fog Delivery Networks (FDN) collaborating to stream videos in order to reduce the latency. Video Streaming Providers cache a base version, which is a type of raw video that allows the generation of all required versions. The base version arrives from cloud and modified versions from base version are processed in the edge (fog) on-demand to users. An important feature of such a proposal is that F-FDN can achieve location-aware caching of video streams. Different of authors' strategy, our approach is data-centric since the edge provider decides to provide the service according application interest, increasing the dynamism of the service provision. Besides, our architecture assumes data stream from different application domains, not limited to video stream.

Authors in [7] proposed a federated file system called iStore. The solution provides a global namespace for unifying geo-distributed edge servers and optimal data placement and analysis in the federation by considering the resource configurations of federated geo-distributed edge servers. The authors' goal is to reduce the time for storing the data and increase the resource utilization across the federated geo-distributed edge servers. Thus, job completion and data migration is completed in more than one edge server. Authors' architecture was designed for solving the problem of the federated file system, while our architecture is more generic since it deals the provision of federated services.

In [13] the authors investigated the business relations between multiple infrastructure providers in a 5G environment. The work considers the creation and upkeep cost of business connections and the price mediation between providers. The model of business relation is based on network formation games where users can use resources located out of the direct reach of their provider. Main goal of authors' work is model the business relations between edge providers, while we propose an architecture where edge providers analyze data stream flow to decide to provide required services, which make the decision taking process dynamic and adaptable to environment changes.

In [5], authors proposed an model for 6G-aware edge federation whose objectives were maximize fog resources and revenue of edge providers, and provide services across the network. Authors formulated a optimization problem of costs and user demands, which was solved by a Stackelberg game algorithm executed in a resource controller in the fog tier. In the proposed architecture, each edge provider decides independently to provision a application service which avoids the need of a central location for resource allocation decision making.

In [2], authors formulated a problem for optimizing the consumed energy and resource provisioning of edge providers. To solve the problem, authors proposed a federated multidimensional fractional knapsack-based method. The proposed method make decision about sharing resource between participating edge nodes in horizontal edge federation system. In the architecture proposed by authors, the horizontal cooperation occurs between edge nodes, which can be located in multiple domains. In our architecture, we assume that edge cooperation occurs between edge providers.

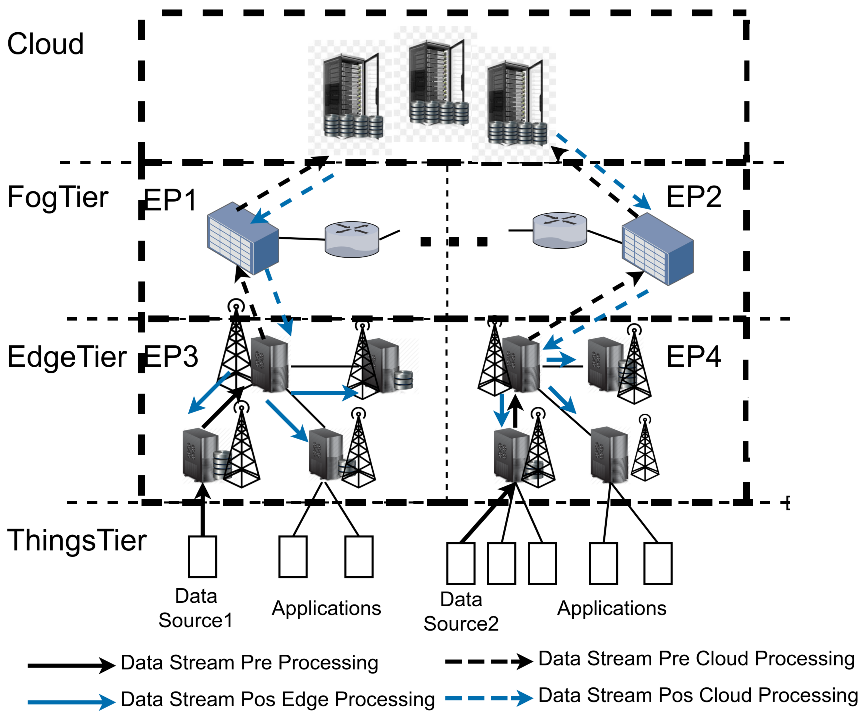
4 System Architecture

In this section, we present the general view of the proposed Edge Federation system architecture. The proposed architecture is composed of four tiers, as shown in Figure 1, namely the Things Tier, Edge Tier, Fog Tier and Cloud. The Edge Provider (EP) is the main element in both the Edge and Fog Tiers. EP corresponds to a set of edge devices or fog devices handled by a unique private edge provider. Two EPs in the Fog Tier can communicate through peering interconnection [13].

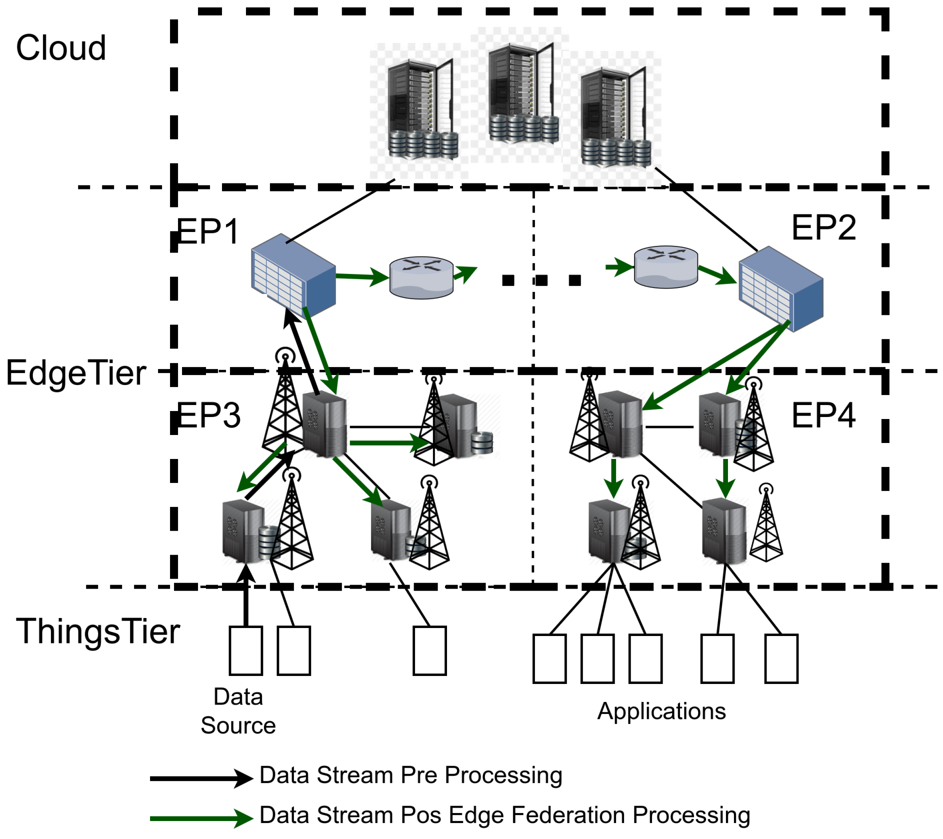
The Things Tier is composed of IoT devices that act as data sources or run user applications. The Edge Tier is composed of edge devices which has direct connection with devices from Things Tier. Nodes from Edge Tier has computing capacity to process data stream generated by applications, but it is limited. The Fog Tier is composed of fog devices, similar to edge devices, but the nodes from Fog Tier does not communicate directly with Things Tier. A Fog Tier EP is a network operator as an Internet Service Provider (ISP) which provides access to the Internet to Edge Tier EP and perform the intermediation between different stakeholders. Finally, Cloud Tier is connected to fog nodes and has abundant resources and huge processing capacity. As the cloud is accessible by all participants, global structures of the federation are public in the cloud.

The Data Stream Flow (DSF) is a sequence of network packets that have the same source and destination network addresses. The DSF is generated in an IoT device called Data Source IoT Device. DSFs can pass through the EPs infrastructure even if the EP is neither origin nor destination of the given DSF. The DSF is generated in a Data Source IoT device which is connected to a Edge Tier EP. This Edge Tier EP, which is directly connected to an Data Source IoT device, is called origin EP. The destination EP is the Edge Tier EP connected to the IoT device running the application that required the DFS. The Edge Tier EP or Fog Tier EP are called middleman EP when the DSF only passes through its infrastructure, but is not neither an origin EP nor destination EP. In Figure 1a, EP3 and EP4 are Edge Tier EP while EP1 and EP2 are Fog Tier EP.

In Figure1a, we observe that DSF generated by the Data Source 1 (black arrow), before processing, passes into the origin EP, which is EP3. The DSF is processed


Cloud
FogTier
EP1
EP2
EdgeTier
EP3
EP4
ThingsTier
Data Source1
Applications
Data Source2
Applications
Data Stream Pre Processing
Data Stream Pre Cloud Processing
Data Stream Pos Edge Processing
Data Stream Pos Cloud Processing

(a) Traditional Edge-Cloud Deployment


Cloud
EP1
EP2
EdgeTier
EP3
EP4
ThingsTier
Data Source
Applications
Data Stream Pre Processing
Data Stream Pos Edge Federation Processing

(b) Edge Federation Deployment

Figure 1: Traditional and Federation Edge Architectures

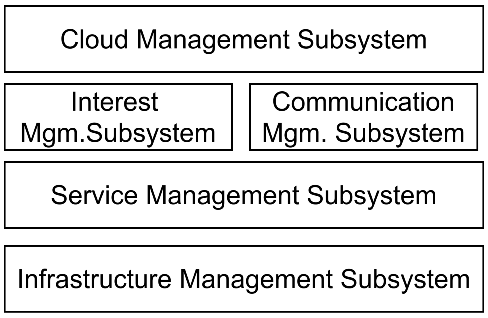

Cloud Management Subsystem
Interest Mgm.Subsystem
Communication Mgm. Subsystem
Service Management Subsystem
Infrastructure Management Subsystem

Figure 2: Main Subsystems of the Architecture

in a edge node from EP3 and the data stream is forwarded (blue arrow) to other edge nodes. Only when necessary, the data stream is forwarded to be processed by cloud (black dashed arrow) and the processed data stream is returned to edge tier (blue dashed arrow).

A similar scenario occurs in EP4, where the Data Source 2 generates the DFS which is processed by a node from EP4 or cloud. The edge computing scenario is advantageous compared to a only cloud approach, but its potential is not entirely explored. For example, scenarios where EP3 is sub-utilized and EP4 is overloaded, EP4 cannot use resources from EP3 for processing application services. EP4 cannot share their applications with EP3 as it has no information about other EPs, thus there is no choice for providing the service in other EP.

Our architecture proposal considers a system based on the federation of EPs. In our system, a Fog Tier EP receiving a DSF can communicate with the cloud or other EP for providing the service in its infrastructure. In Figure 1b, EP1 detects the DSF and communicates to the cloud or EP4 for processing the DSF. EP1 deploys the required services in its infrastructure and the DSF does not pass by cloud anymore. To make the Edge Federation possible, we assume that each EP needs to participate in a federation and implement a set of software components, which are presented in the Section 5.

4.1 Architecture Subsystems

Our Edge Federation Architecture consists of five subsystems: Infrastructure Management, Service Management, Communication Management, Interest Management and Cloud Management. Subsystems are defined by Edge Federation, i.e. by the consortium of all edge providers pertaining to Edge Federation. Figure 2 shows the Edge Federation subsystems.

Infrastructure Management is a subsystem whose function is to monitor the network and other physical resources, besides allocating and deallocating resources. This subsystem continuously analyzes all data streams passing by EP. When a new data stream is detected, this subsystem triggers the service provision of EP.

Service Management subsystem decides to initiate the service provision when either a user request a service or a new data stream is detected. The service provision for a data stream occurs when there is low utilization of edge provider resources. The Service Management communicates to the origin edge provider that is generating the data stream. From the origin network address, Service Management knows the origin edge provider, allowing to establish the connection with the origin edge provider.

The Service Management subsystem also is responsible to provide services to applications. It is fed by the Infrastructure Management subsystem with important information about the state of the edge provider infrastructure. Descriptions of services stored in edge provider local cache or global database are accessed by this subsystem.

The Communication Management subsystem is responsible to establish the communication between two distinct edge providers. Client and server brokers compose this subsystem and an edge provider runs both client and server. An edge provider that is the end destination of the data stream or a middleman edge provider can trigger a client broker to communicate to the server of the origin edge provider to require the authorization for providing services of the application generating that data stream. An origin edge provider runs a server broker to meet requests from client brokers of external edge providers.

The Interest Management is a subsystem whose goal is to map interests to services, making interests available to the edge provider. This subsystem interacts with the Service Management subsystem to obtain a set of services from an interest. A service set description is retrieved from either an interest local cache of the edge provider or a global database of interests. The Edge Federation is responsible to define the global database of interests and the service set description.

The Cloud Management subsystem stores public and global storage of the federation. Besides, this subsystem provisions a component to delivery data stream for users.

5 Detailed Architecture

In this section, we describe the detailed architecture and subsystem components (see Figure 3).

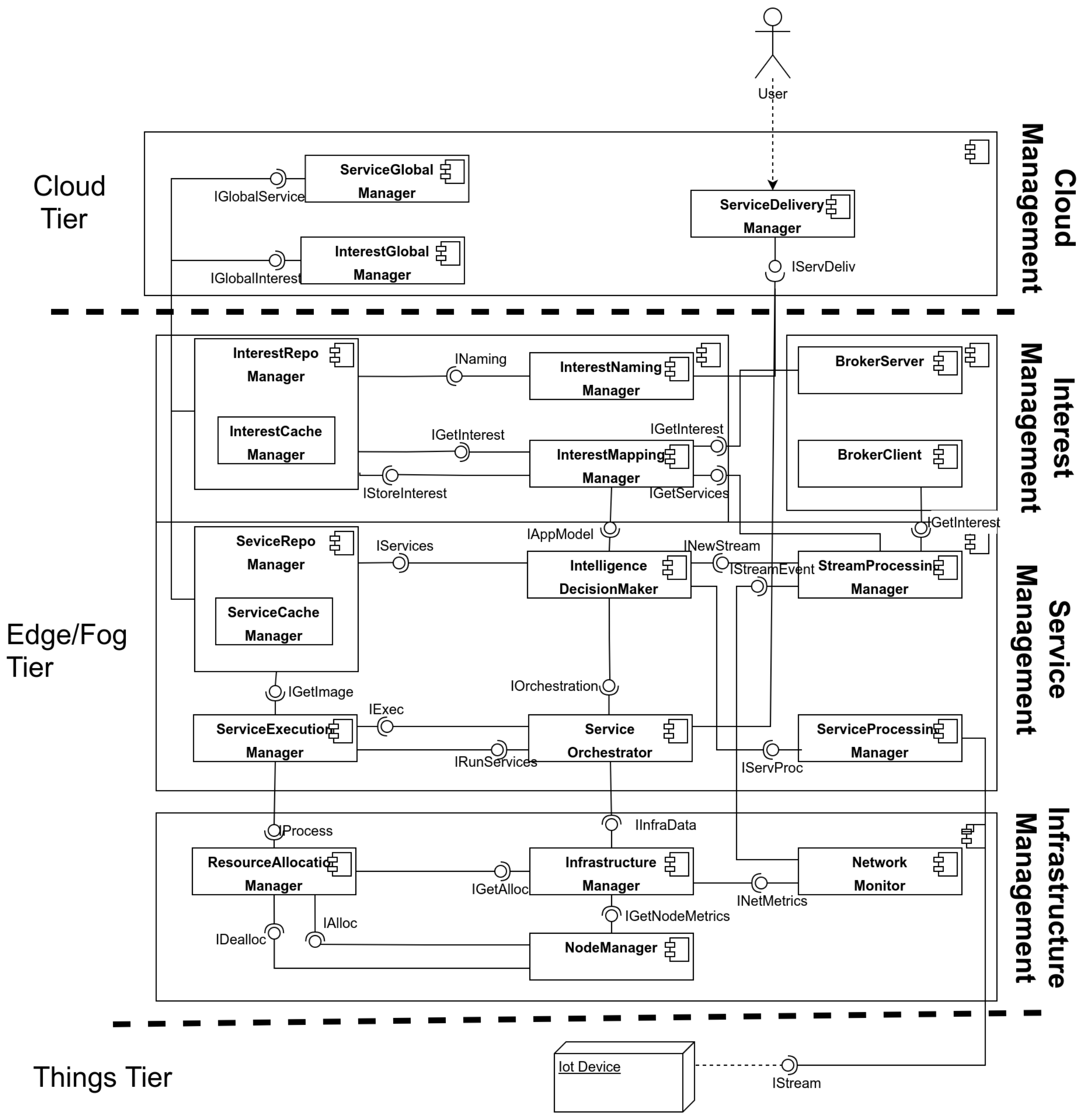


Figure 3: Detailed Architecture

5.1 Infrastructure Management Subsystem

Infrastructure Management is the subsystem responsible to analyze uninterruptedly the traffic passing by the network, manage the infrastructure resources and receive or process the data stream generated by the application.

5.1.1 NetworkMonitor

The *NetworkMonitor* (NM) is the component that continuously monitors the DSF that passes through EP. The EP can be either a DSF destination or a middleman actor of DSF, i.e. an EP who has direct links to other providers but is not directly interconnected with a business link generating that data stream [13]. In the latter case, DSF is traffic passing through the network, which augments the required bandwidth for EP. NM has a interface *INetMetrics*, which provides network metrics to *InfrastructureManager* component, and *IStreamEvent* which notifies the *StreamProcessingManager* component that a new data stream event was detected.

5.1.2 InfrastructureManager

The *InfrastructureManager* (IM) component has the global view of the EP infrastructure. IM stores updated information about resource usage of processing node resources and communication links: CPU, memory, storage and link bandwidth. The resource usage information is obtained through *IGetNodeMetrics* interface. IM also obtains from *IGetAlloc* interface information about the allocation of virtualized resources, i.e. containers. The *IInfraData* interface provides entire information about the resources.

5.1.3 ResourceAllocationManager

The *ResourceAllocationManager* (RAM) component monitors, handles, allocates and deallocates resources of the EP infrastructure for executing services. RAM is responsible to handle the containerization system and provide information about allocated resources. *IAlloc* and *IDealloc* interfaces allow RAM allocating and deallocating resources of a node.

5.1.4 NodeManager

The *NodeManager* component handles the CPU, memory and bandwidth resources of the node. It provides node metrics through *IGetNodeMetrics* and receives commands for allocating and deallocating resources through *IAlloc* and *IDealloc* interfaces.

5.2 Service Management Subsystem

Service Management Subsystem has the objectives of deciding to process services associated with a data stream of a external EP and managing services assigned to EP.

5.2.1 StreamProcessingManager

StreamProcessingManager (SPM) is the component that receives a new data stream notification from *NetworkMonitor*, through *IStreamEvent* interface, and has the responsibility to get the data stream information. This component discovers the demand, i.e. interest and services, associated with the data stream and forwards to *IntelligenceDecisionMaker* the demand. SPM component obtains the interest from the *IGetInterest* interface. Interest and associated services are obtained from the origin EP, via *BrokerClient* communication. The *IGetInterest* interface returns to SPM the interest associated to the data stream obtained from *IStreamEvent*. The *IGetServices* interface provides the service description from *InterestMappingManager*. The *InterestMappingManager* has the view of all databases of interests made available by the Edge Federation. The obtained service descriptions are forwarded to *IntelligenceDecisionMaker* through *INewStream* interface.

5.2.2 Intelligence Decision Maker

The *IntelligenceDecisionMaker* is the main component of the architecture since it decides if EP will provide the services required for processing a new detected data stream. *IntelligenceDecisionMaker* is notified by *StreamProcessingManager* component, through *INewStream* interface, about a new data stream detection.

When a new stream is detected, *IntelligenceDecisionMaker* needs to decide if it is worthwhile to provide the services and process the detected stream. The problem solved by *IntelligenceDecisionMaker* is to minimize the network traffic passing by its infrastructure while ensuring that provided services keep respecting the application requirements.

To take the decision, initially *IntelligenceDecisionMaker* obtains the application model description (AMD) from *IAppModel* interface. From AMD, *IntelligenceDecisionMaker* gets the stored services from *ServiceRepoManager* component accessing the *IServices* interface. On following, *IntelligenceDecisionMaker* orders the *ServiceOrchestrator* component, via *IOrchestration* interface, to choose the resources for allocating and deploying the services.

5.2.3 ServiceOrchestrator

The service orchestration is responsibility of the *ServiceOrchestrator* (SO) component, which chooses processing nodes for executing services. This component obtains infrastructure metrics and information about running services for determining where executing services. *IInfraMetrics* and *IRunServices* interfaces provide infrastructure metrics and running service information to SO, respectively. The SO component also is required to orchestrate services arriving from user end. *IServProc* and *IServDeliv* interfaces notify SO component when new data stream of a user requires service processing.

5.2.4 ServiceExecutionManager

The *ServiceExecutionManager* (SEM) component receives the order from *ServiceOrchestor* and processes the required services. While *ServiceOrchestrator* chooses processing nodes for executing services, SEM gets stored services for executing them. SEM receives the service processing request from *IExec* interface with the service information and their configurations. Stored services are obtained through *IGetImage* and *IProcess* interface is called to run the service. The *IRunServices* is a interface for making available the information about running services.

5.2.5 ServiceRepoManager

EP has a database abstraction handled by the *ServiceRepoManager* component. This component provides the *IGetImage* interface to make available stored service images to *ServiceExecutionManager*. Service images can be stored either locally or remotely in the cloud. Images stored locally are handled by *ServiceCacheComponent*, while images stored in the cloud are handle through *IGlobalService* interface. Services are locally stored to avoid obtaining it from the *ServiceGlobalManager*, which is a component implemented in the cloud.

5.2.6 ServiceProcessingManager

The *ServiceProcessingManager* component receives a DFS from IoT devices connected to edge nodes of EP. This component obtains the stream from *IStream* interface and forwards it to the *IntelligenceDecisionMaker* through *IServProc* interface.

5.3 Communication Management Subsystem

Conventionally, system architectures employ intermediate brokers that orchestrate the communication with external entities [1]. The Communication Management subsystem promotes the communication between two different EPs and consists of components: *BrokerClient* and *BrokerServer*. Upon receiving a DSF, the *BrokerClient* of a middleman EP requires authorization from the *BrokerServer* of an origin EP for providing the services associated with the DSF. When the EP origin authorizes the middleman EP to meet the service set , the origin EP *BrokerServer* sends to *BrokerClient* the interest associated with the data stream. Thus, *BrokerClient* receives the interest and the middleman EP can provide the required services.

5.3.1 BrokerClient

The *BrokerClient* (BC) component is responsible to initiate the communication between a middleman EP and a origin EP. Upon receiving a DSF from the origin EP, the middleman EP needs to communicate to the origin EP for getting the interest associated with the DFS. The BC component makes the communication possible, offering the *IGetInterest* interface which returns the interest obtained from the origin EP.

5.3.2 BrokerServer

The *BrokerServer* (BS) is a component that meets the requests from some BC from an external EP for the service provision. When a request arrives in the *BrokerServer*, it queries the *InteresMappingManager* component, through *IGetInterest* interface, to obtain the interest and the application description related to the generated data stream. The interest information and application description for a data stream are then provided in response to the *BrokerClient*.

5.4 Interest Management Subsystem

The Interest Management Subsystem has the function of creating, storing and mapping interest and an Application Model Description (AMD). The AMD is the representation of the required services and their relationships. The EP needs an AMD to know if it can process the DSF arriving into EP.

Interest creation consists of defining a unique interest name to an AMD. The interest storing consists of the maintenance of a interest repository, which is stored locally in EP or made available globally in the cloud. Finally, the main function of *InterestManagementSubsystem* is to map, when required, interests to an AMD.

5.4.1 InterestMappingManager

The *InterestMappingManager* component is the responsible component to search for the interest and their associated services. The main objective of *InterestMappingManager* is to translate an interest to an AMD.

The *InterestMappingManager* interacts with the *InterestRepoManager*, through *IGetInterest* interface, for obtaining the required interest. The required interest is stored in either the EP, locally, or the cloud, externally. The *InterestMappingManager* also interacts with components responsible for managing the services of the EP from the Service Management subsystem.

5.4.2 InterestRepoManager

The *InterestRepoManager* stores or searches for interests. An interest is stored locally, through the *InterestCacheManager* component, or acessed remotely. For searching for a interest, firstly, *InterestRepoManager* searches for the interest locally, when the interest has already been requested previously. When the interest has never been requested by EP, then *InterestRepoManager* communicates with a global database of interest through the *IGlobalInterest* interface.

The *InterestCacheManager* is a component that abstracts the local interest database and provides a standard protocol to obtain a AMD from an interest.. The local interest database is a permanent storage memory, maintained locally by the EP, which contains all interests and their respective AMD. The local interest database is intended to provide quick access to an interest that has already been previously requested, and therefore is already known by the EP.

5.4.3 InterestNamingManager

The main objective of the *InterestNamingManager* component is to provide interest naming for an application stream. When the user requests to initiate a stream processing, the application requests EP to create a new interest providing an AMD. The main function of *InterestNamingManager* is to map an interest to an AMD. The interest-AMD mapping is stored through *INaming* interface. The *InterestNamingManager* registers the application interest in a global database of interests, enabling external EP to provide the services required by his application.

5.5 Cloud Management Subsystem

The Cloud Management Subsystem has two components to store services and interest in a global database of interests on the cloud. Besides, it has a component whose function is to delivery content to the user called *ServiceDeliveryManager*.

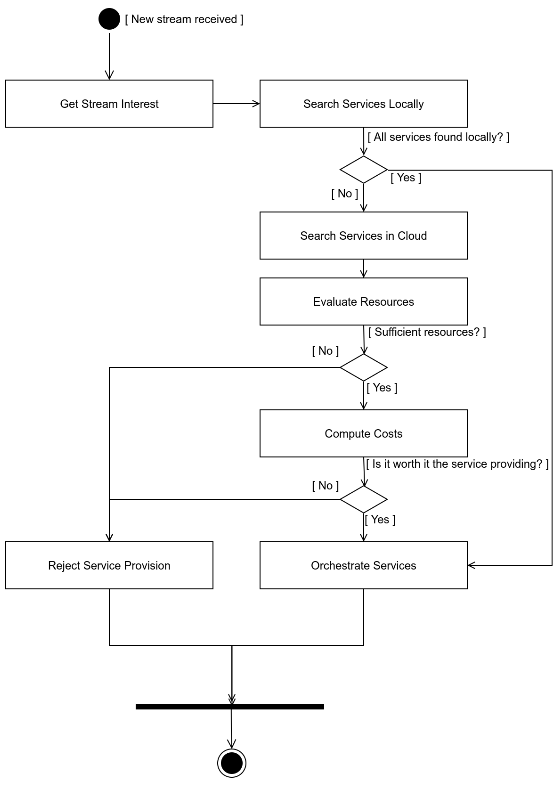


Figure 4: Overview of Intelligence Decision Maker operation process

Regarding the service and interest storage, the components are *InterestGlobalManager* and *ServiceGlobalManager*. The main function of the *InterestGlobalManager* component is to search for an interest in the global database of interests, which is a cloud storage that maintains interests. It is a Software as a Service (SaaS) maintained by the Federation or a third-party. Members of the Edge Federation define their specifications and definitions. The *InterestGlobalManager* is an abstraction for the interest database.

ServiceGlobalManager is a database that contains services provided by all EPs that make up the Edge Federation. When an EP has never served a given service, it needs to get it from *ServiceGlobalManager* to provide the service for the first time.

6 INTELLIGENCE DECISION MAKER

In this section, we described the operation of the main component of the architecture. After detecting a DSF, the *IntelligenceDecisionMaker* component should decide if it is worth providing services for processing the DSF.

To make the decision, the resources and costs associated with data stream processing are analyzed.

Figure 4 depicts the operation process of the *IntelligenceDecisionMaker* component. Initially, the component receives a new DSF and communicates with the origin EP to obtain the interest associated with such a DSF. Services associated with the interest are locally searched and if all services are found, then the *ServiceOrchestator* component is required for distributing services in the edge nodes of the EP. In the case some service is not found locally, it is obtained from the interest global database in the cloud.

After obtaining all services, the *IntelligenceDecisionMaker* component evaluates the infrastructure resources to know if available resources are sufficient to meet all service requirements. The resource evaluation consists of computing the following resources: number of processing cores, number of instructions, available memory, and bandwidth. If there are not enough available resources, then the data stream continues to be propagated into the EP infrastructure, but no service is provided.

When the EP has sufficient resources to provide the service, the cost associated with the service provisioning is computed. This cost corresponds to the price offered by the origin EP to accept the service to be provided by the middleman EP. The price of service provision is dynamically hired between EPs during the communication between client and server brokers. If the offered price covers the service provisioning costs, then the EP can provide the service.

When the price does not cover the middleman EP cost, services are not provided. Otherwise, *IntelligenceDecisionMaker* calls the *ServiceOrchestrator* to orchestrate the services in edge nodes of the infrastructure.

7 Final Remarks and Ongoing Work

The Edge Federation architecture was developed to distribute the processing of large-scale data streams across multiple edge providers. It is intended to promote collaboration and distribute the service provision between different edge providers.

In our data-centric architecture, the service provisioning is guided by required data by the application. In the proposed architecture, the application requires data as a service, and the service is executed in edge nodes with the same interest as the application. The interest representation allows mapping application service to edge nodes located in multiple edge domains.

In future work, we intend to evaluate the performance of the Intelligence Decision Maker in terms of cost, resource utilization, and application requirements. The cost and resource utilization investigation aims to provide relevant information from infrastructure for the edge provider. With this information, the edge provider can decide when provisioning an application service is more beneficial than merely use the infrastructure to forward network traffic.

Acknowledgements

This work is partially funded by FAPESP (grant 2015/24144-7 and 2020/12891-0). Flavia Delicato and Paulo Pires are CNPq Fellows.

References

[1] F. M. Awaysheh, M. Alazab, M. Gupta, T. F. Pena, and J. C. Cabaleiro, "Next-generation big data federation access control: A reference model," *Future Generation Computer Systems*, vol. 108, pp. 726–741, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19315948

[2] H. Baghban, C.-Y. Huang, and C.-H. Hsu, "Resource provisioning towards opex optimization in horizontal edge federation," *Computer Communications*, vol. 158, pp. 39–50, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419319796

[3] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X15003015

[4] X. Cao, G. Tang, D. Guo, Y. Li, and W. Zhang, "Edge federation: Towards an integrated service provisioning model," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, p. 1116–1129, Jun. 2020. [Online]. Available: https://doi.org/10.1109/TNET.2020.2979361

[5] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Stackelberg game for service deployment of iot-enabled applications in 6g-aware fog networks," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5185–5193, 2021.

[6] A. Karkouch, H. Mousannif, H. Al Moatassime, and T. Noel, "Data quality in internet of things: A state-of-the-art survey," *Journal of Network and Computer Applications*, vol. 73, pp. 57–81, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804516301564

[7] A. Khan, M. Attique, and Y. Kim, "istore: Towards the optimization of federation file systems," *IEEE Access*, vol. 7, pp. 65 652–65 666, 2019.

[8] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S235286481730247X

[9] D. Mocrii, Y. Chen, and P. Musilek, "Iot-based smart homes: A review of system architecture, software, communications, privacy and security," *Internet of Things*, vol. 1-2, pp. 81–98, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660518300477

[10] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[11] Z. Sharmin, A. W. Malik, A. Ur Rahman, and R. MD Noor, "Toward sustainable micro-level fog-federated load sharing in internet of vehicles," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3614–3622, 2020.

[12] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[13] L. Toka, A. Recse, M. Cserep, and R. Szabo, "On the mediation price war of 5g providers," *Electronics*, vol. 9, no. 11, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/11/1901

[14] V. Veillon, C. Denninnart, and M. A. Salehi, "F-fdn: Federation of fog computing systems for low latency video streaming," in *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, 2019, pp. 1–9.

[15] Y. Zhao, W. Wang, Y. Li, C. Colman Meixner, M. Tornatore, and J. Zhang, "Edge computing and networking: A survey on infrastructures and applications," *IEEE Access*, vol. 7, pp. 101 213–101 230, 2019.

[16] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

AUTHOR BIOGRAPHIES

Tiago C. Xavier received his Doctorate degree in Computer Engineering in 2020 from the Federal University of Rio de Janeiro. He is currently a postdoctoral researcher at Universidade Federal Fluminense. His research interests include distributed systems, Internet of Things, Machine Learning and Wireless Sensors Network, Edge and Cloud Computing.

Flávia C. Delicato is an Associate Professor at the Fluminense Federal University, Brazil. Her primary research interests are IoT, adaptive middleware and Edge Computing. She has published 2 Books and over 160 papers. She serves as Associate Editor of the Ad Hoc Networks, ACM Computing Surveys, IEEE Transactions on Services Computing Journals, and Area Editor of the IEEE Open Journal of the Communications Society.

Paulo F. Pires is an Associate Professor at the Fluminense Federal University, Brazil. His main research interests are at the intersection of Software Engineering and Distributed Systems. He has published more than 200 articles and has four patents registered with the USPTO (United States). He is currently Associate Editor of the IEEE Open Journal of the Communications Society and member of the editorial board of the International Journal of Computer Networks (CSC Journals).