UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

cemisid
centro de estudios en microelectrónica y sistemas distribuidos

Middleware para IoT basado en Analítica de Datos

Jose Aguilar
Dpto. de Computación, Facultad de Ingeniería

Noviembre 2018

UNIVERSIDAD COMPLUTENSE MADRID

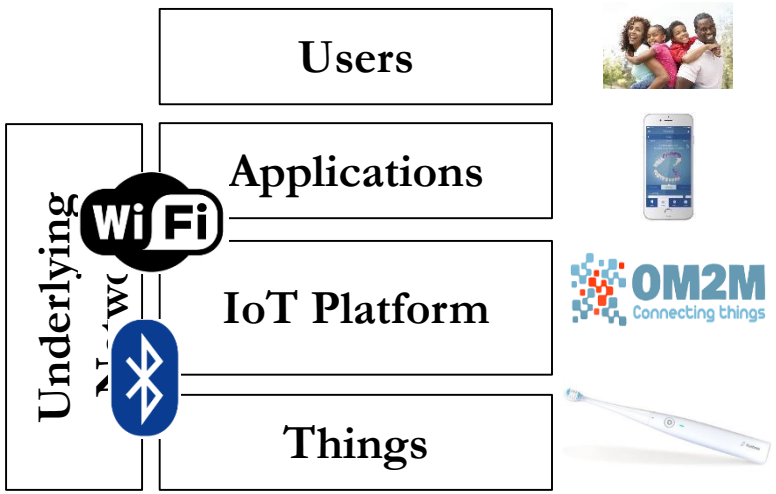
Agenda

- ➔ Context
- ➔ Problem
- ➔ Our general approach : An autonomic cycle for QoS provisioning
- ➔ Our Contributions
  - A Classification OR clustering model for Diagnostic?
  - Flyweight Network Functions
  - Virtualization of Transport-level Functions and Protocols
  - Internet traffic classification
- ➔ Perspectives

1. Internet of Things (IoT): extension of the Internet to communicating "objects" other than computers (e.g. : sensors, actuators, ...)

2. IoT High-Level Architecture entities:

For instance :

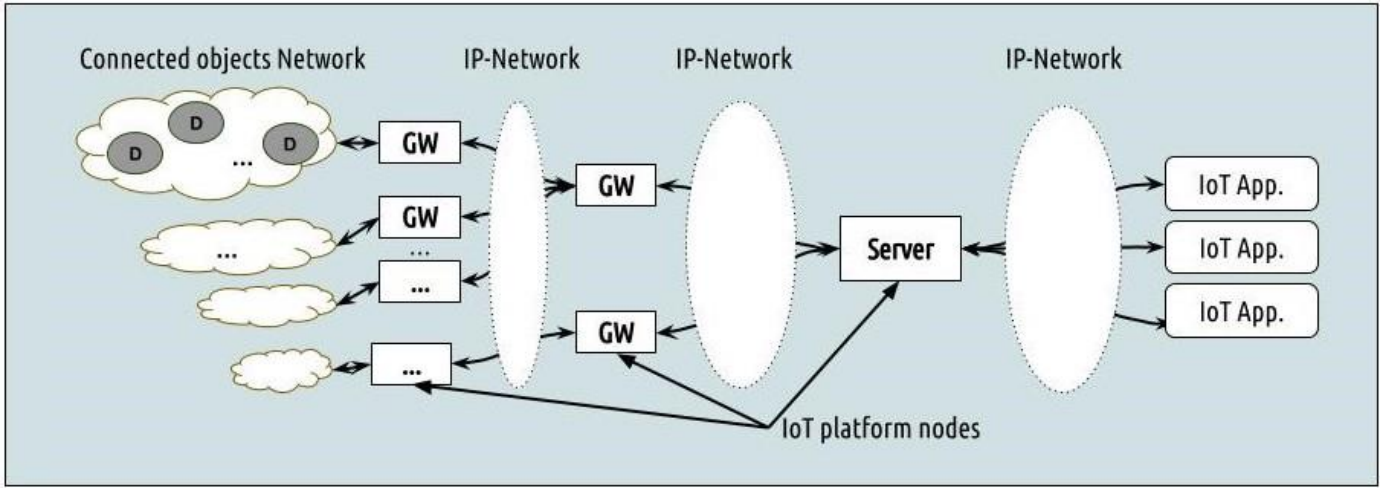


3. The IoT applications: Smart Cities, Self-Driving Cars, Smart Factories, *eHealth*, etc.

4. Tens of billions of connected things within a few years [1].

[1] Leading the IoT Gartner Insights on How to Lead in a Connected World, 2017

The reference architecture for IoT [1]:



[1] ETSI TS 102 690 V1.1.1 “Machine-to-Machine communications (M2M); Functional architecture”, october 2011, p15

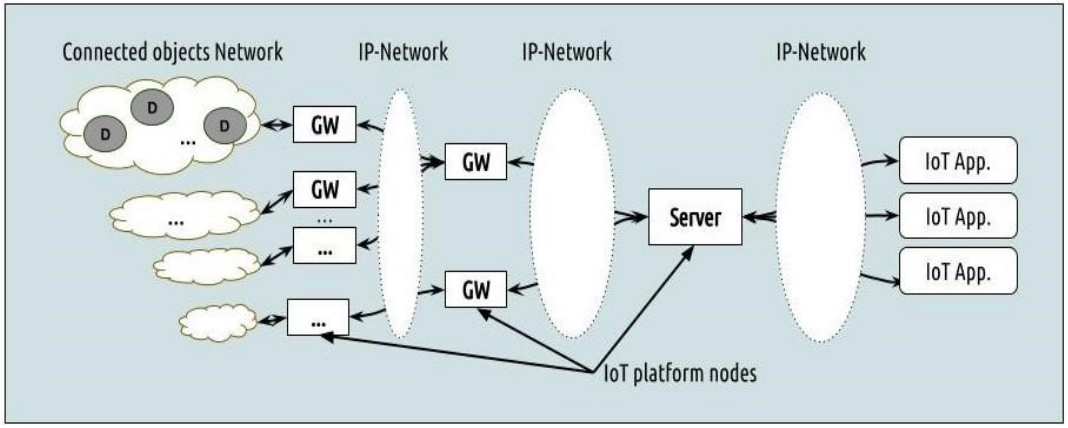
1. IoT applications and their QoS requirements (bounded response time, availability, etc.)

Example of an application's QoS requirements (Traffic Signal Violation Warning Requirements [3])

- Communication from infrastructure-to-vehicle
- Transmission mode: periodic
- Minimum frequency (update rate): ~ 10 Hz
- Allowable latency ~ 100 msec

2. Two bottlenecks facing QoS :

- at the level of IP networks
- at the level of IoT Platform nodes.



[3] The CAMP Vehicle Safety Communications Consortium, DOT HS 809 859, "Vehicle Safety Communications Project Task 3 Final Report Identify Intelligent Vehicle Safety Applications Enabled by DSRC", May 2004.

On the QoS management in IoT, 3 families of approaches can be found in the literature

- Approaches that ensure QoS at the MW (middleware) level for applications. These approaches consider MW as a bottleneck and use **mechanisms to differentiate the services** offered by MW.

[A] Q. Han, and N. Venkatasubramanian, "Autosec: An integrated middleware framework for dynamic service brokering," IEEE distributed systems online, 2(7), 2001, pp. 518-535.
[B] F. C. Delicato, et al., "Reflective middleware for wireless sensor networks," Proceedings of the 2005 ACM symposium on Applied computing, March 2005, pp. 1155-1159
[C] M. Sharifi, M. A. Taleghan, and A. Taherkordi, "A middleware layer mechanism for QoS support in wireless sensor networks," Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, April 2006, pp. 118-118
[D] A. Agirre, et al. "QoS aware middleware support for dynamically reconfigurable component based IoT applications," International Journal of Distributed Sensor Networks, 12(4), 2016.

- Approaches that are using MW to provide application QoS through the **reconfiguration of the underlying network**. These approaches do not consider the MW as problematic but rather as a tool to overcome the problem of the network.

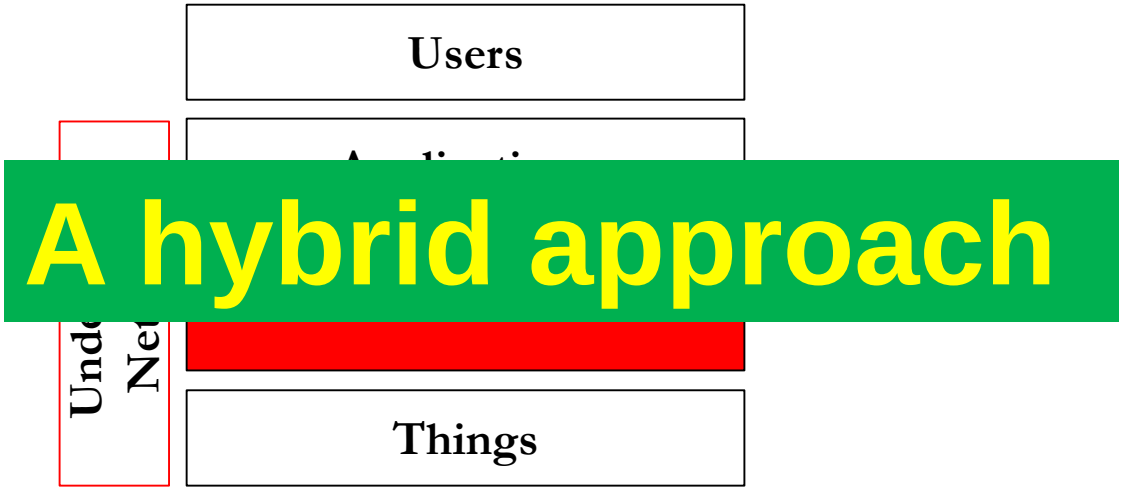
[E] W. Heinzelman, et al., "Middleware to Support Sensor Network Applications," Network, IEEE, vol. 18, issue 1, 2014, pp. 6-14
[F] S.-Y. Yu, Z. Huang, C.-S. Shih, K.-J. Lin, J. Hsu, "QoS Oriented Sensor Selection in IoT System," IEEE and Internet of Things (iThings/CPSCom), September 2014, pp. 201-206
[G] J. R. Silva, et al., "PRISMA: A publish-subscribe and resource-oriented middleware for wireless sensor networks," Proceedings of the Tenth Advanced International Conference on Telecommunications, July 2014, pp. 8797.

- **Hybrid** Approaches

[H] F. C. Delicato, et al., "Reflective middleware for wireless sensor networks," Proceedings of the 2005 ACM symposium on Applied computing, March 2005, pp. 1155-1159
[I] N. Hua, N. Yu, and Y. Guo, "Research on service oriented and middleware based active QoS infrastructure of wireless sensor networks," 10th International Symposium on Pervasive Systems, Algorithms, and Networks, December 2009, pp. 208-213

Considering the QoS : 2 bottlenecks

- The IoT platform
- The underlying network



IoT High Level Architecture (HLA)

Existing solutions

Several solutions have also been proposed that address the QoS issue for IoT contexts:

- Based on **service differentiation**: processing the requests differently, depending on their priority.
- The QoS mechanisms are provided at the **initialization** of the platform.
- Inadequate when a **service is nonexistent** on a node/when the computing **resources are insufficient**.
- Tactile Internet is a new concept where **this limitation is very important**.

We need dynamic and autonomic solutions

QoS-oriented mechanisms

- The approach we are exploring is to **dynamically provide the middleware with mechanisms** that allow it to maintain its performance closer to the application requirements. We call these mechanisms: **QoS-oriented mechanisms**.

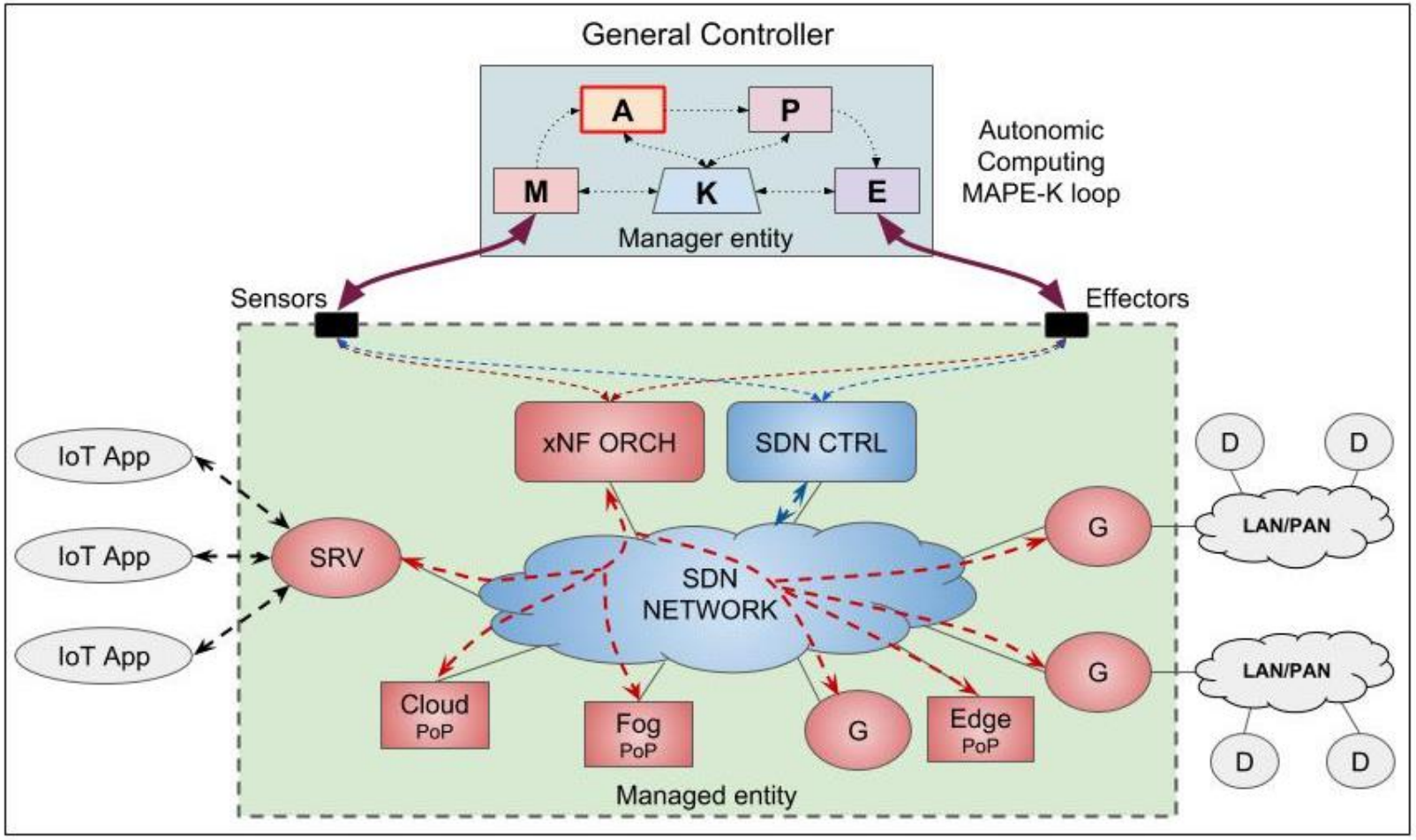
- 2 kinds of mechanisms can be considered:

  - **Traffic-oriented** (inspired from the network layer): Traffic Marker/shaper, Message or Task scheduler, etc.

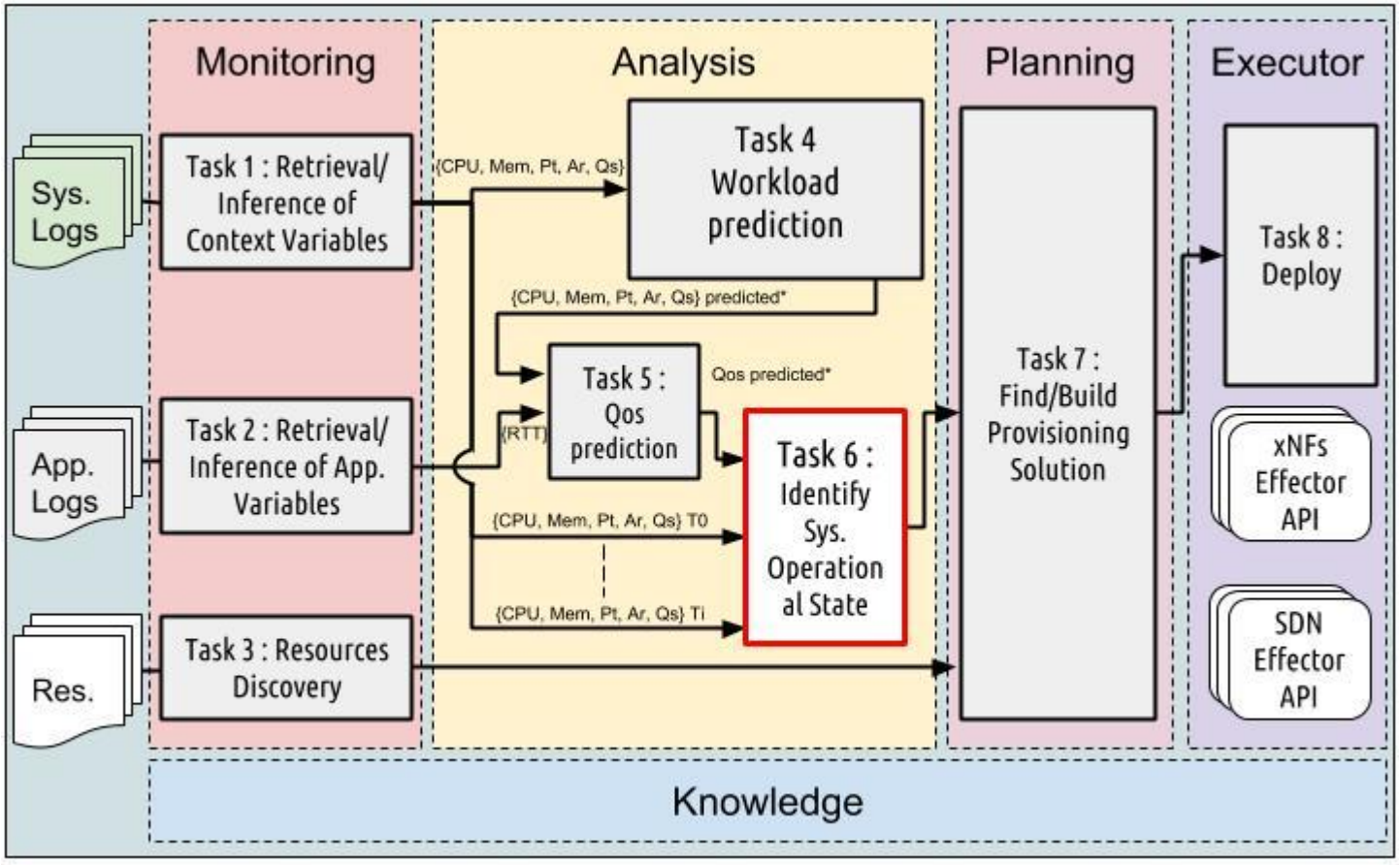
  - **Resource-oriented** (inspired from cloud computing): scale in/out mechanism, load balancers, replication and so on, etc.

HLA Model for a Dynamic and Autonomic System

A **hybrid approach** in a heterogeneous environment:

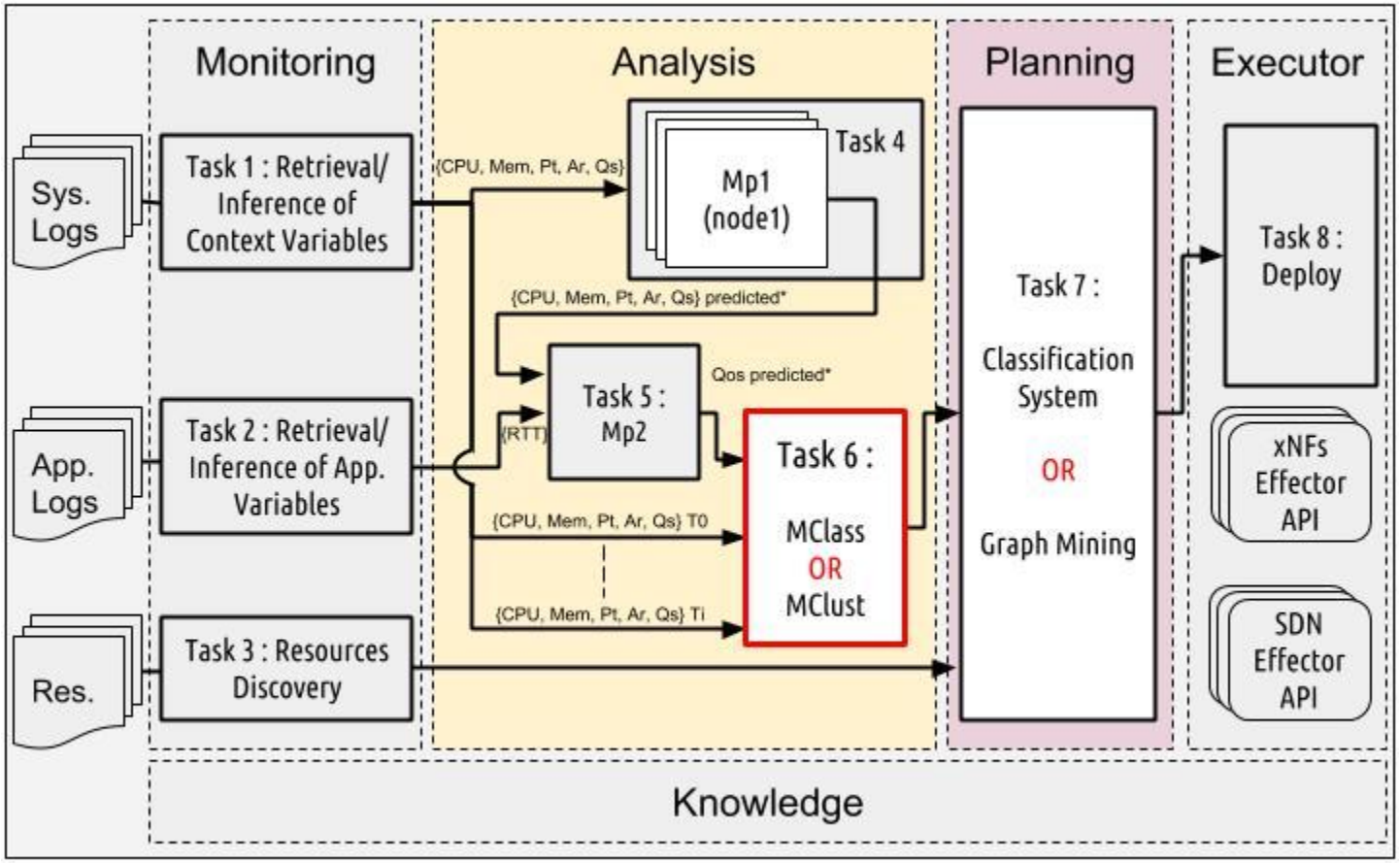


An autonomic cycle for QoS provisioning



A distributed control system inspired by the MAPE-K loop

A Classification OR clustering model for Diagnostic?



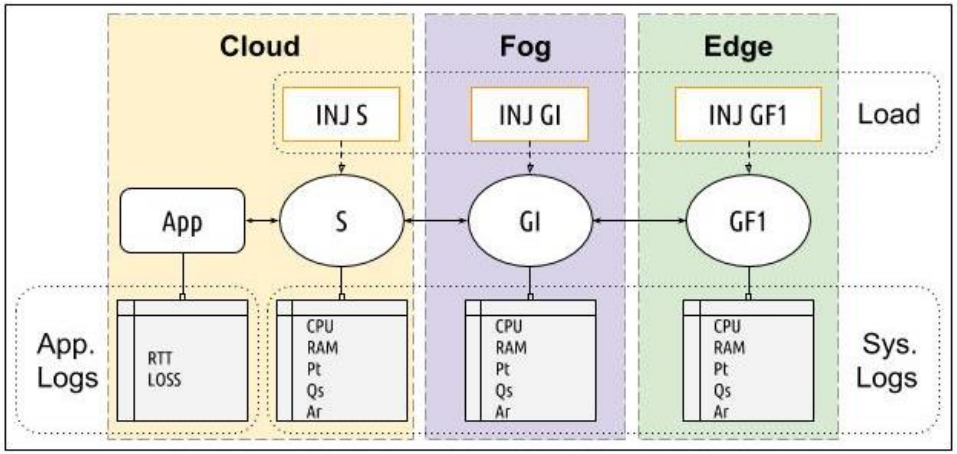
A Classification OR clustering model for Diagnostic?

- **Classification**
  - Classifying data into **predefined** categories
  - It requires expertise to identify these categories in advance.
  - Model required in the planning phase: a classification system that **associates identified categories with actions**.

- **Clustering**
  - Grouping data into a set of clusters according to a given **similarity metric**
  - Model required in the planning phase: a model that **discover in real-time the set actions** to be executed for the current cluster

A Classification OR clustering model for Diagnostic?

Scenario



Operational state of the IoT platform

N	Cloud	Fog	Edge	App0
1	Not loaded	Not loaded	Not loaded	3 req/sec
2			Loaded	
3		Loaded	Not loaded	
4			Loaded	
5	Loaded	Not loaded	Not loaded	
6			Loaded	
7		Loaded	Not loaded	
8			Loaded	

A Classification OR clustering model for Diagnostic?

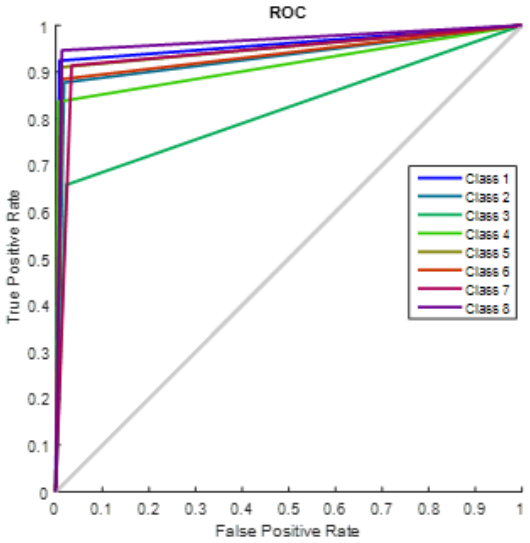
- **Experiments and Result Analysis : Classification with LAMDA-HAD**

Performance metrics for the classifier

Accuracy	Precision	Recall	F-meas.	Sens.	Spec.	AUC
0,8740	0,8507	0,8678	0,8574	0,8678	0,9746	0,9212

Performance metrics for the classifier by eliminating descriptor 13

Accuracy	Precision	Recall	F-meas.	Sens.	Spec.	AUC
0,8436	0,7977	0,8429	0,8153	0,8429	0,9601	0,9015



ROC metric of the classification model

LAMDA = Learning Algorithm Multivariable and Data Analysis

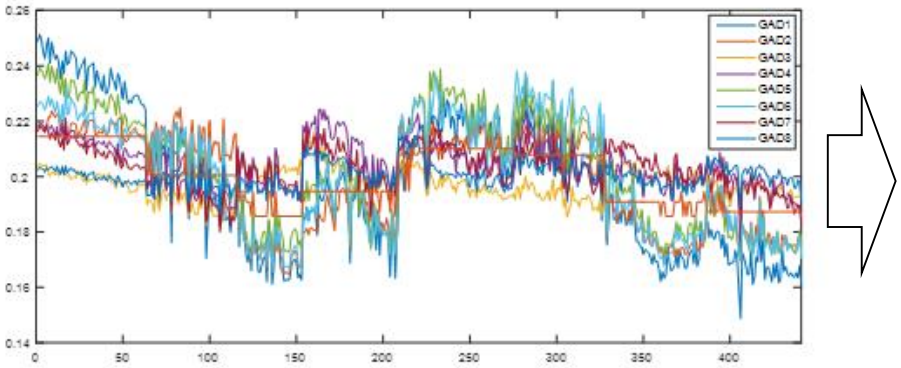
A Classification OR clustering model for Diagnostic?

- **Experiments and Result Analysis : Clustering with LAMDA-RD**

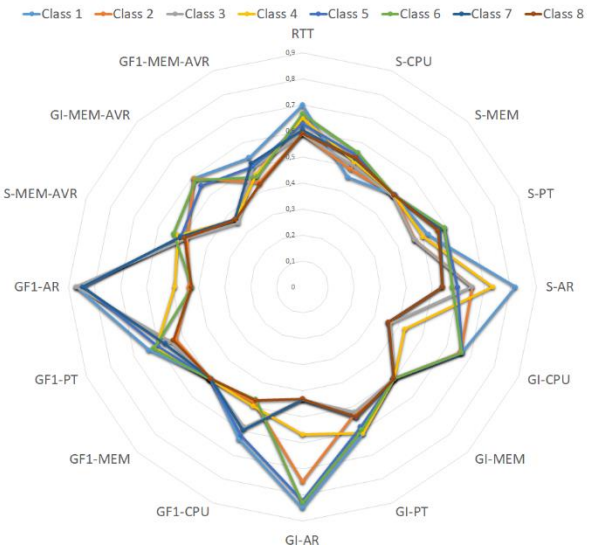
Performance metrics of clustering algorithm

	SC	SSW	SSB	SSW/SSB	CHC	# CLUST.
REAL CLUSTERS	-0.1497	**0,6120**	**0,4979**	**1,5128**	387.1738	8
LAMDA RD	**-0,0261**	0,6848	0,3672	1,8649	**293,7209**	15

Profile of each Cluster/Class



LAMDA result example

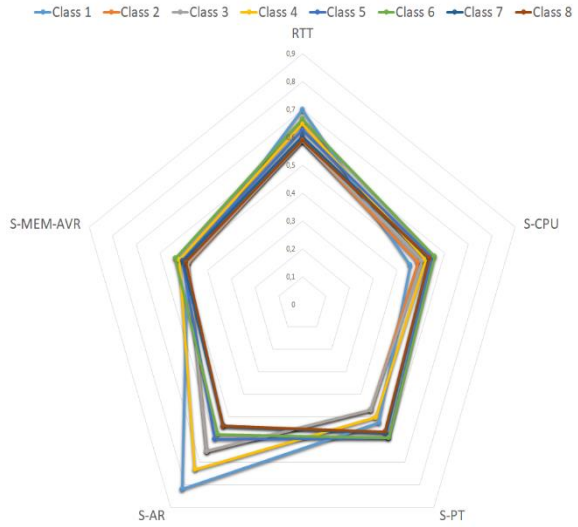


General Profile of the IoT platform with 16 descriptors

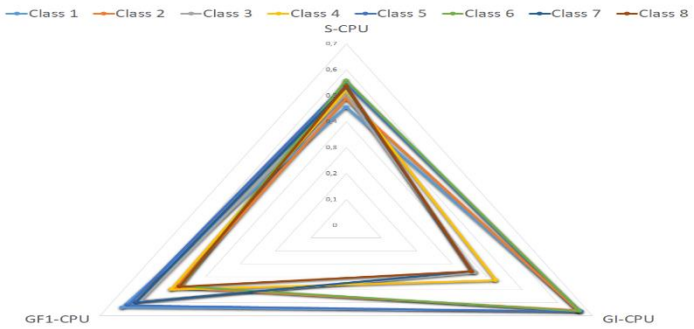
LAMDA = Learning Algorithm Multivariable and Data Analysis

A Classification OR clustering model for Diagnostic?

- *General profile of the IoT Platform*
- Profile by entity
- Profile with specific descriptors (e.g. CPU and/or RAM fo the entities)
- ...



Profile of the server in the IoT platform



Profile of the CPU descriptor in the IoT platform

A Classification OR clustering model for Diagnostic?

- **Knowledge models:**
  - The classification model is quite robust, because it is capable of **determining the operations states** in the system.
  - For the clustering algorithms, they **are affected by the characteristics of the descriptors**, giving good results, but not better than those obtained with the classification.

- **The cycles:**
  - based on the classification model, the **dependence of experts is greater**, both to label the operational states (data) and to determine the tasks aimed at improving the platform.
  - based on the clustering model, although the results are not so good, not depending on the expert, which **gives more robustness to the process**, but it requires **the interpretation of the clusters**.

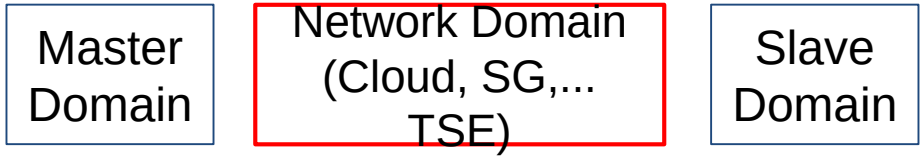
- The clustering algorithms **give more clusters than the number of operational states**, which is important to analyze, because **they can be sub-states within states, which the experts do not know**. That could improve the QoS results

Flyweight Network Functions

Instantiation of the autonomic cycle

> **Case Study of IoT context :** In the case of IoT, the managed entity is an IoT (traditional) platform composed of several entities, namely a cloud Server(S), Intermediate Gateways(IG) and Final Gateways(FG).

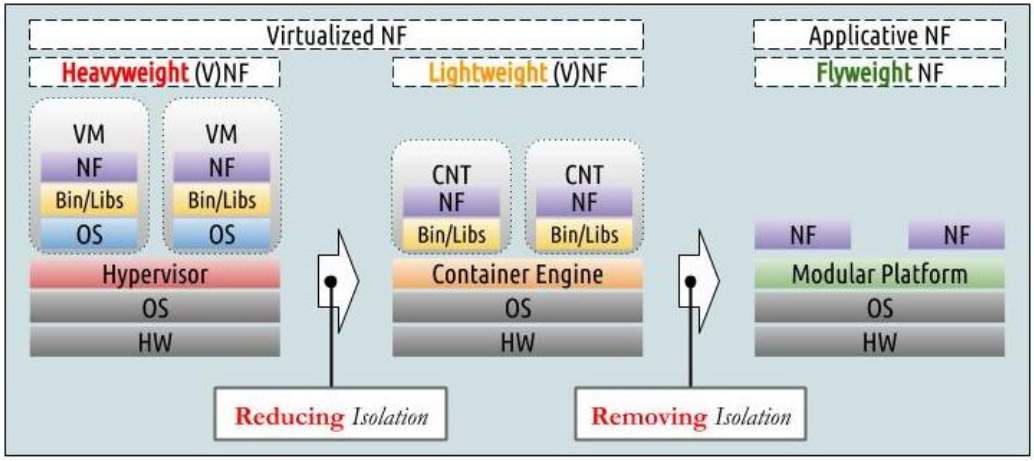
> **Case Study of Tactile Internet :** In this case study, the managed entity to consider is the "network domain" which is composed by several entities such as Cloud, serving gateway(SG) and Tactile Support Engine (TSE).



Flyweight Network Functions

Presentation of the fNF concept

Definition : Flyweight NF are deployable network functions in the form of software modules.



Formally, a fNF is defined as an instantiation of NF having the following properties:

- is the instantiation of a NF in the form of a software module without virtualization overhead;
- is an implementation of a NF without isolation in the User space, just like an application;
- is dynamically deployable / deletable / editable/ configurable;
- is instantiable on a compatible platform for fNFs deployment, typically a modular framework.

What a fNF is not:

- a VNF, because it is not instantiated as a virtualization container (CNT/VM) but as a software module;
- a PNF, because it is not instantiated on hardware built for this unique (dedicated) use.

Network slicing concept

ITU-T [7] definition: a network slice as a logical network that provides specific network capabilities and network characteristics. The (network) slicing consists in building slices on demand.

- Initially thought to share resources on a communication infrastructure.
- Now more and more considered to **perform QoS provisioning**.
- Instantiation of network slices done with **VNFs via VM/CNT**.

Limits of the existing slicing implementations:

- Virtualization containers induces a **virtualization overhead** [8] potentially problematic for some IoT deployment targets (e.g. RPi used as IoT gateway) with very limited resources.
- Some NF, by their size, utility, to be instantiated in the form of **VNF can be counterproductive**.
- This instantiation method of NF does **not cover heterogeneity of the future 5G networks**.

⇒ The concept of network slicing (as it is conceived currently) is **based on cloud-type infrastructure and will be hardly usable to achieve end-to-end slices**, i.e. connecting data producers and consumers.

[7] ITU, Terms and definitions for IMT-2020 network, 2017. Recommendation ITU-T Y.3100.
[8] Z. Li, M. Kihl, Q. Lu, et al., “Performance overhead comparison between hypervisor and container based virtualization,” in Advanced Information Networking and Applications (AINA), 2017 IEEE 31st International Conference on, pp. 955–962, IEEE, 2017.

Flyweight Network Functions

A highlighted feature in **current VNF platform deployments** (justifying the use of VMs/CNTs) is the isolation so that NFs running on the same (physical) hardware do **not interfere with each other** from two standpoints [9]: **security and performance**.

In IoT, when it comes to we claim that this feature can be **discussed and ignored**:

- **Security**: when the slice provider is the only actor capable of building slices, the burden of protecting the source code and the traffic of the NFs will be guaranteed upstream by integrity verification techniques and encryption of the traffic.
- **Performance**: the management of the overall performance of the slice will make it possible to balance the expected ”characteristics” by taking into account the workload of NFs hosts.

Removing the isolation techniques between NF

- we lose: level of security, performance guarantee;
- we win: removal of the overhead (resource, deployment time, etc.), reduction of the complexity of the host platform of NFs, increase of the possible number of hosts of NF.

⇒ Under certain conditions/domains, we propose the concept of fNF in order to allow network slicing for IoT.

Flyweight Network Functions

A Framework of Modular Flyweight Network Functions

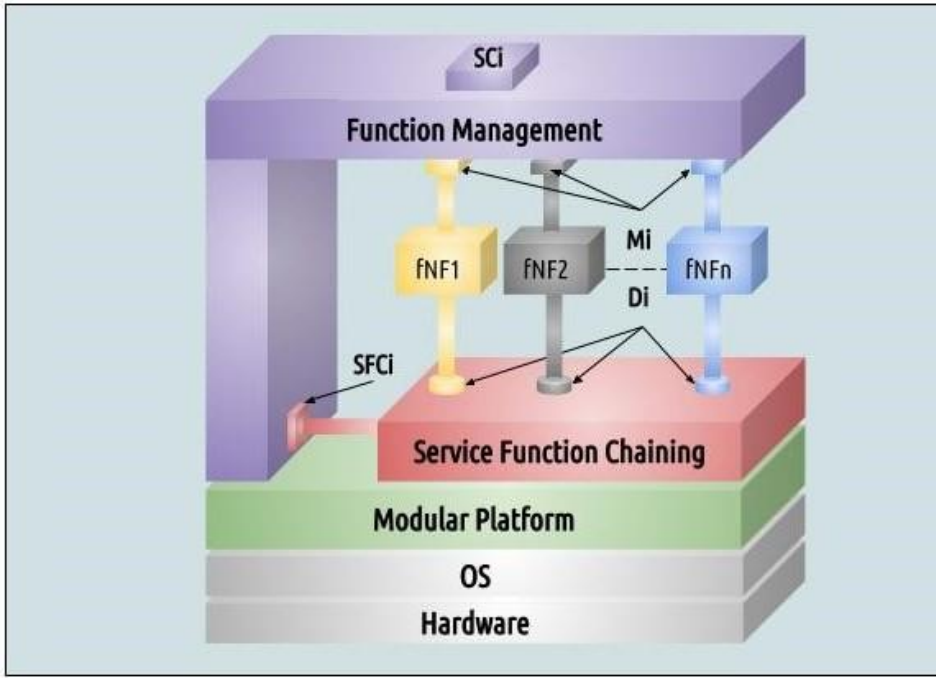
Considering the 3GPP model of NF [10], we propose the following architecture for the implementation of fNFs

- **Function Management**: functionalities needed to configure fNFs. Its role is to configure the fNFs and the Service Function Chaining component with the slicing policy it has received from the Slice Controller through the service controller interface (SCi)

Our Autonomic Cycle

- **Service Function Chaining** (SFC) deals with the interconnection of fNFs between them. It performs this task based on the configuration received from the Function Management;

- **Modular Platform** is the fNFs execution platform; it implements a complete and dynamic component model.



Flyweight Network Functions

Network Slices provisioning: Our Autonomic Cycle

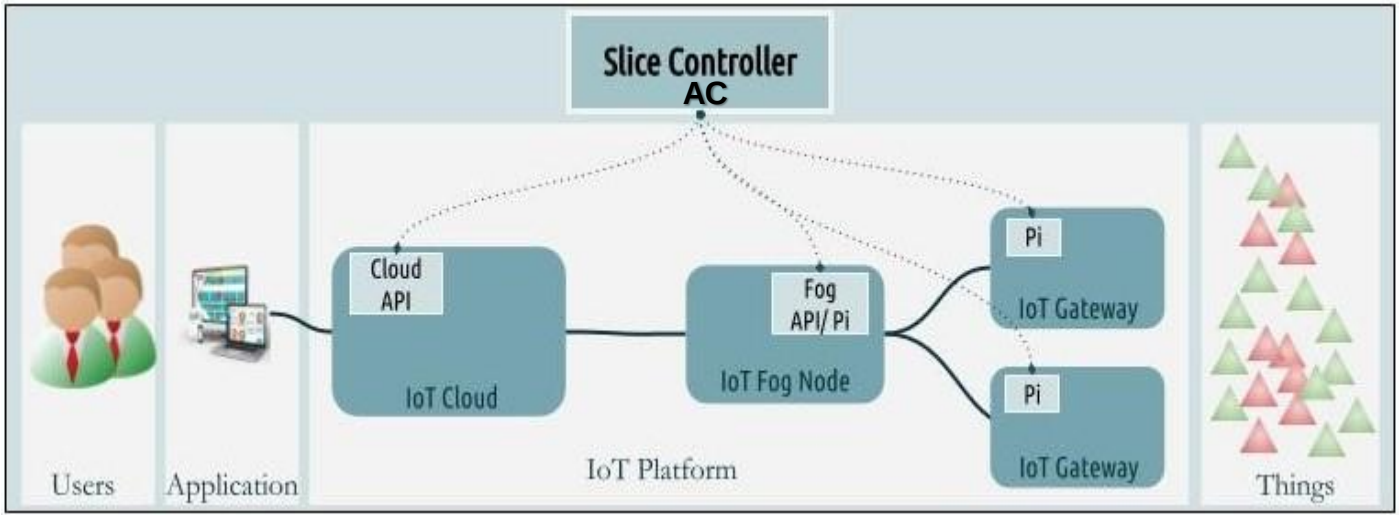
To provision a Slice:

- retrieves the QoS requirements of IoT applications;
- analyzes these needs and choose a network service consisting of V/fNFs with properly defined characteristics;
- packages NFs according to hosts that can meet the desired characteristics;
- instantiates these V/f NFs on the selected hosts;
- implements the policy associated with the network service, i.e.: (i) configures the deployed fNFs; and (ii) configures the policy associated with the slice on the SFC.

Flyweight Network Functions

Example

A user, request a given platform level slice. Our AC, through a set of successive tasks, sets up this slice using VNFs, but also fNFs.



The requested slice has the following functional and non functional (i.e. QoS oriented) characteristics:

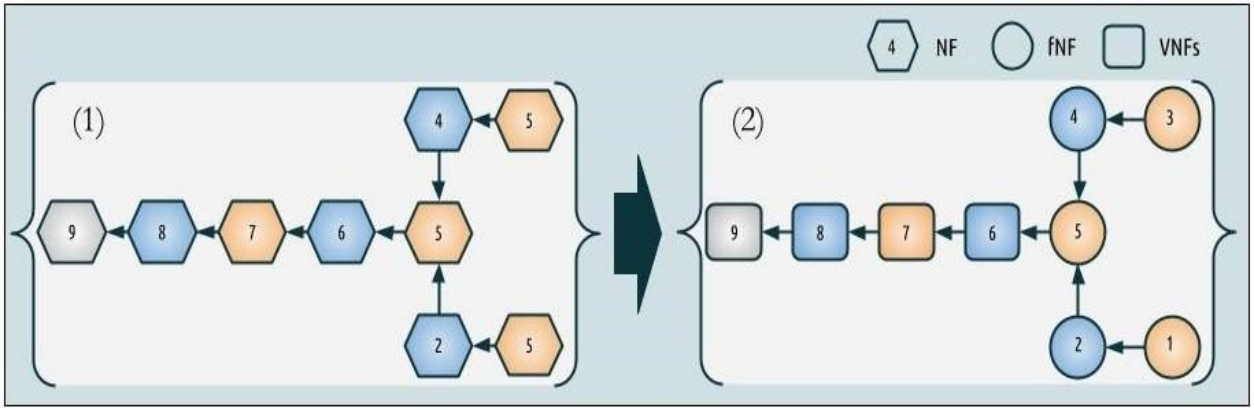
- allowable latency: 10ms
- availability: 90%
- services: Data Collection, Stream processing, Data Storage
- service life: 7h.

Flyweight Network Functions

Slice construction

Step 0 & 1: Upon receipt of the user's request, the AC selects in a service catalog the network service (NS) to offer for such a request.

This NS is composed of nine NFs : four brokers, four stream processors, and a database.



Step 2: The AC then packages the selected NFs into VMs or CNTs (for VNF) and Components (for fNF).
AC completes the NS with the associated packages information :

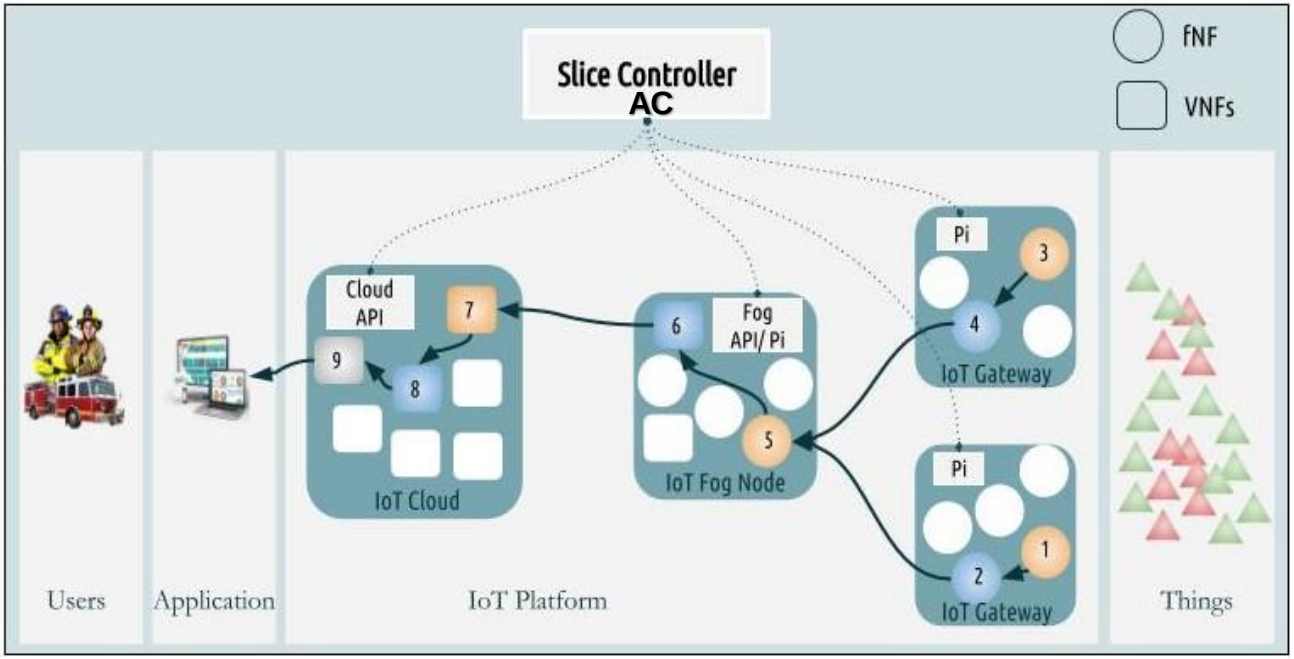
- each gateway: an fNF broker and an fNF Stream processor,
- the Fog node: an fNF broker and a VNF Stream processor,
- the Cloud: a VNF broker, a VNF Stream processor and VNF Storage.

Flyweight Network Functions

Slice construction

Step 3: Once the NS is built, the V/fNFs are deployed on the selected hosts.

Step 4: At the end of the deployment, the V/fNFs are configured with the slicing policy associated with the NS. The slice is then ready to be used, and a positive response is sent to the user having requested the slice.

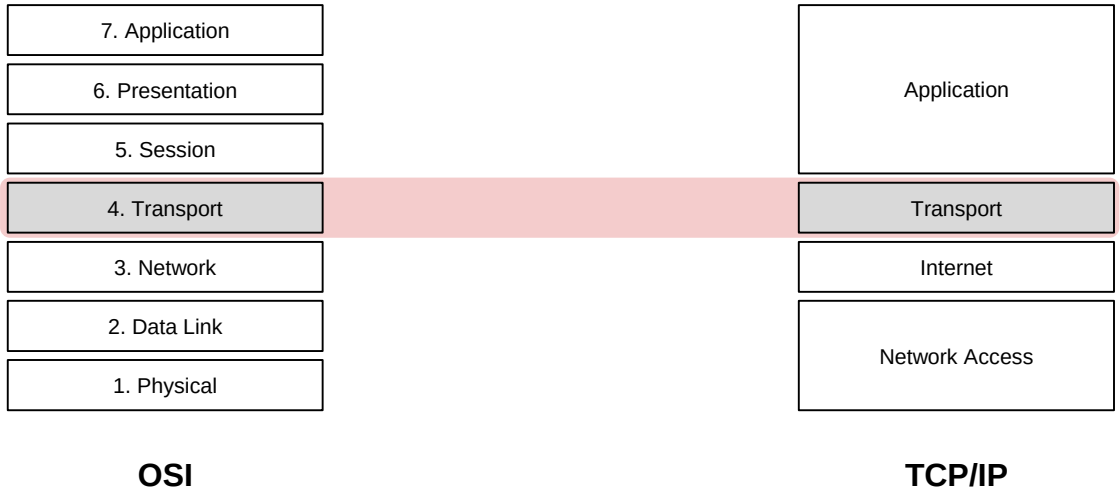


Virtualization of Transport-level Functions and Protocols

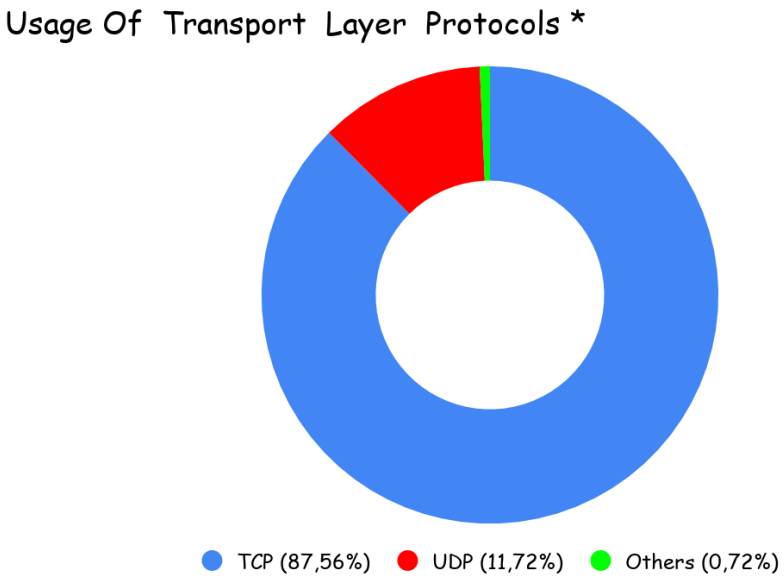
Considering the QoS : 2 bottlenecks

- The IoT platform ⇒ main problematic
- **The underlying network**

Internet architecture is layered around two popular standards: OSI and TCP/IP.



Virtualization of Transport-level Functions and Protocols



TCP and **UDP**

are the most used Transport protocols.

***** D. Murray et al, “An Analysis of Changing Enterprise Network Traffic Characteristics”, 2017.

But, TCP and UDP are limited…

Virtualization of Transport-level Functions and Protocols

Two directions of research to overcome Transport issues :

Proposition of **single protocol**:

- by extending TCP features (MPTCP, …)
- by building a new one:
  - from scratch (SCTP, DCCP, …)
  - on top of UDP (QUIC, …)

Redesign the **entire Transport layer**:

- TAPS, the IETF group,
- NEAT, H2020 project.

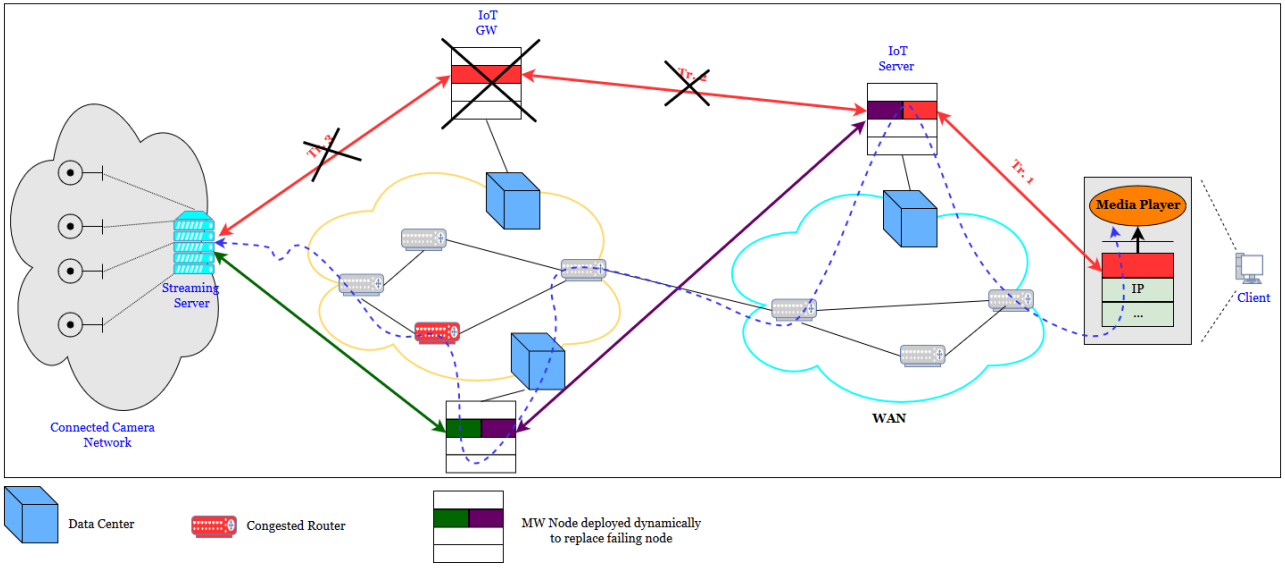
At the same time, emergence of new paradigms:

- Software Defined Networking (SDN)
- Network Function Virtualization (NFV)

Virtualization of Transport-level Functions and Protocols

Our goal…

Dynamic and *timely* deployment of a Transport component.



Two main techniques in our approach:

- ***Virtualization***: following ETSI-NFV* standard principles.
- ***Modularization***: around the idea of *Transport Functions*

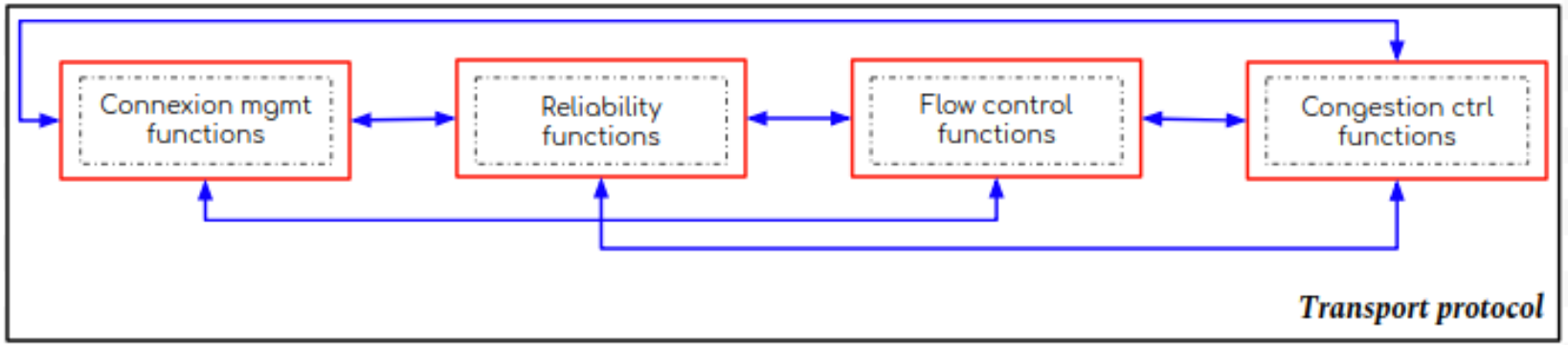
* European Telecommunications Standards Institute.

Virtualization of Transport-level Functions and Protocols

Every Transport ***protocol*** is implementation of a set of basic functions...

Packaging of each function within virtualization Container...

Dynamic ***protocol*** construction by connexion of container containing basic functions.



Our Autonomic Cycle

Architecture control: ***Transport Function Manager*** (TFM)

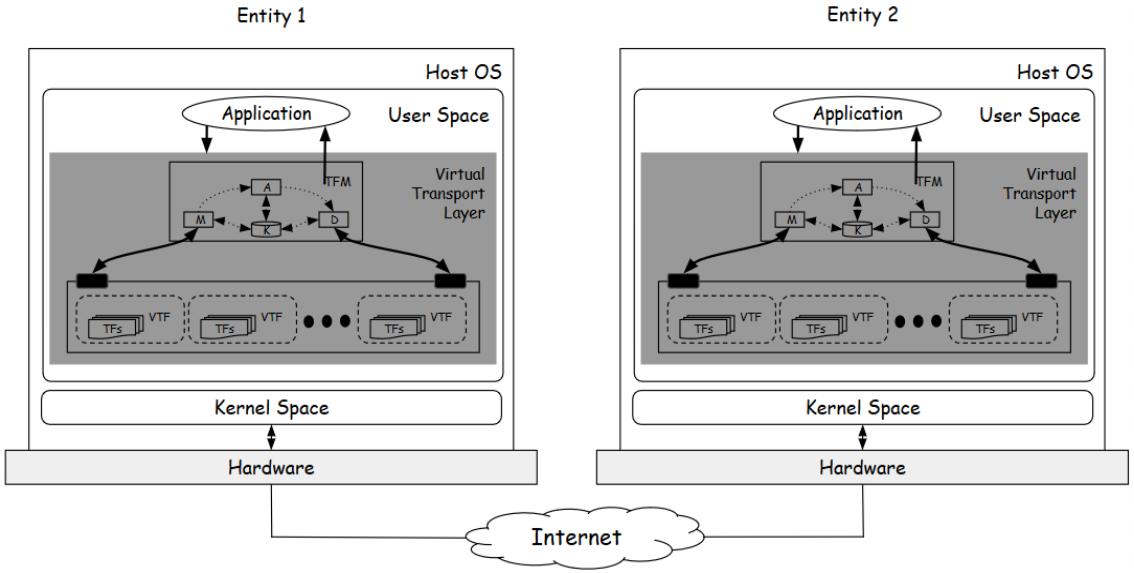
01

Specification and formalisation of ***Transport Function*** (TF)

02

Construction of ***Transport Services*** (TS) and ***Transport Protocol*** (TP) from TFs

Virtualization of Transport-level Functions and Protocols



Overview of TFM and its components

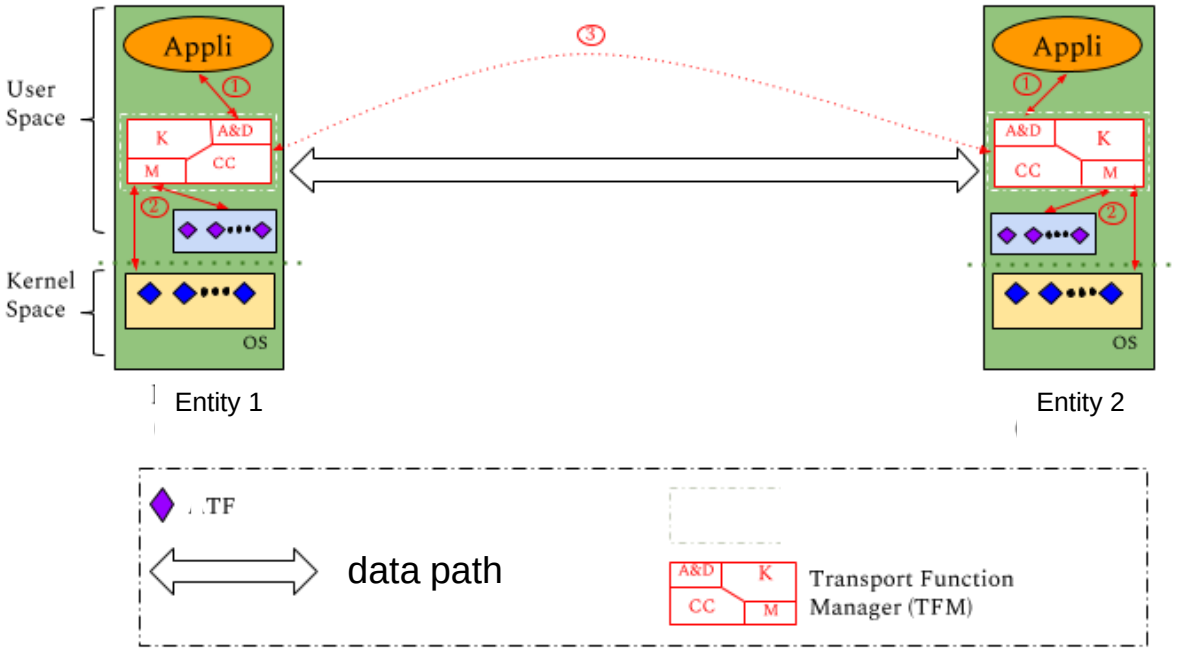
Our Autonomic Cycle

***Transport Function Manager* (TFM):** a distributed control architecture that aims at dynamically build TS and deploy all necessary TFs to provide the required TS.

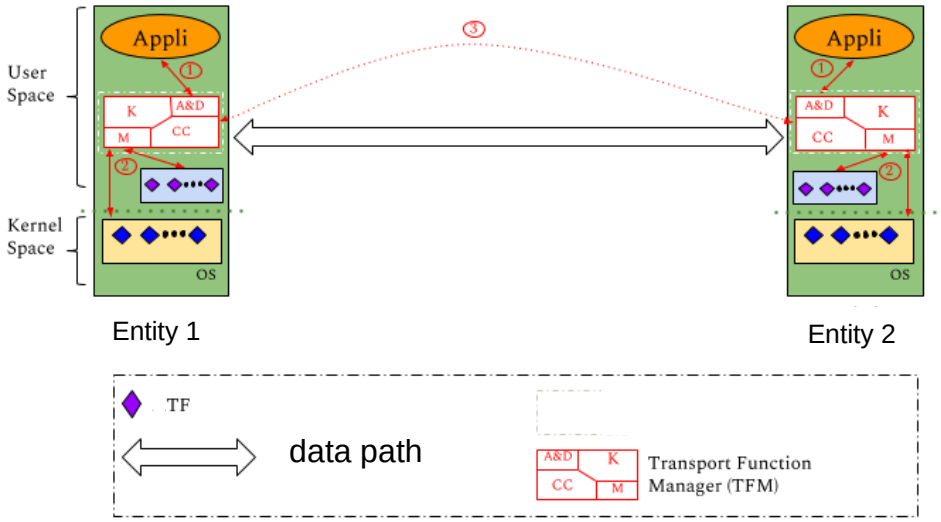
Virtualization of Transport-level Functions and Protocols

TFM is a **distributed control system inspired by the MAPE-K loop (our Autonomic Cycle)**.

The TFM is deployed in a virtualization container (VM or CNT) local to the entity, or kernel space of the entity, involved in the data exchange.



Virtualization of Transport-level Functions and Protocols

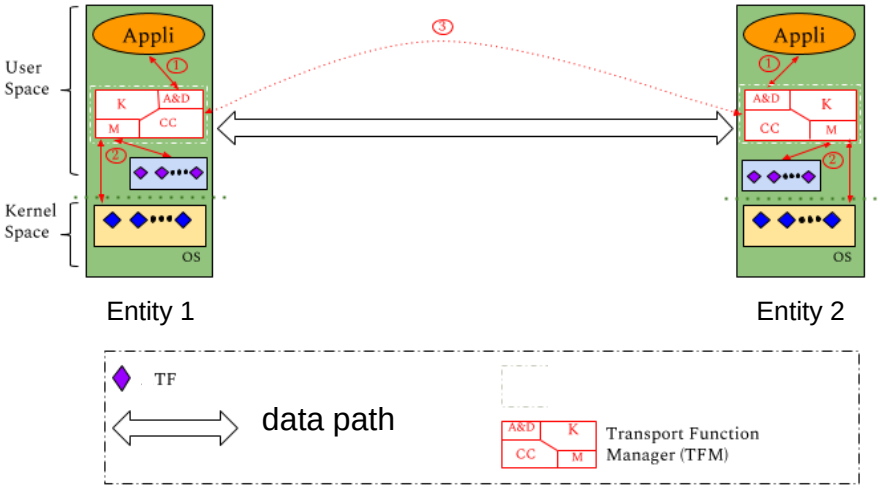


Monitoring (2): A component that collects the characteristics of the Transport and the host OS to determine:

- If ETP or equivalent exists
- If the kernel supports KTF deployments
- The TFs deployed on the entity

Knowledge: Knowledge base allowing the TFM to store the information collected by the Monitoring. TFs are managed using a graph to discover TS (Transport Service)

Virtualization of Transport-level Functions and Protocols



Analysis & Decision (1): Intercepts the service requests of the hosting entity's apps, and using the knowledge base (K), is able to:

- to browse the graph of TFs to build the TS,
- and when there are missing TFs, execute the **deployment decision algorithm**
  - UTF: user space support it
  - KTF: authorization of the OS
  - VTF otherwise

TF graph: a mixed graph $G = (V, A, E)$ where:

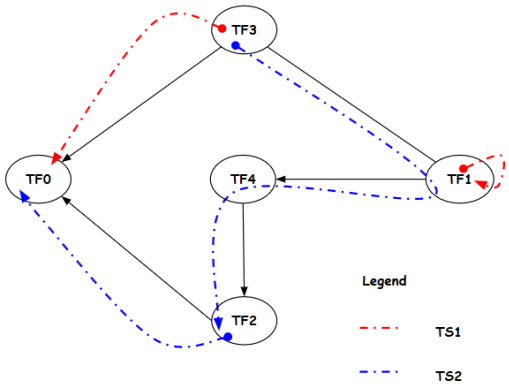
- $V = \{TF_1, TF_2, \ldots, TF_n\}$: set of TFs deployed on entity
- A, represents dependencies between TFs
- E, represents unordered relations between TFs

TF Graph:

- Used for the description of a protocol by a set of Transport Services (TS)
- A TS is *dynamically* built and validated through a TFs graph
- A TS is defined by two paths:
  - one way: the TFs execution order in Rx
  - 2nd way: the TFs execution order in Tx

Virtualization of Transport-level Functions and Protocols

TF Graph illustration:

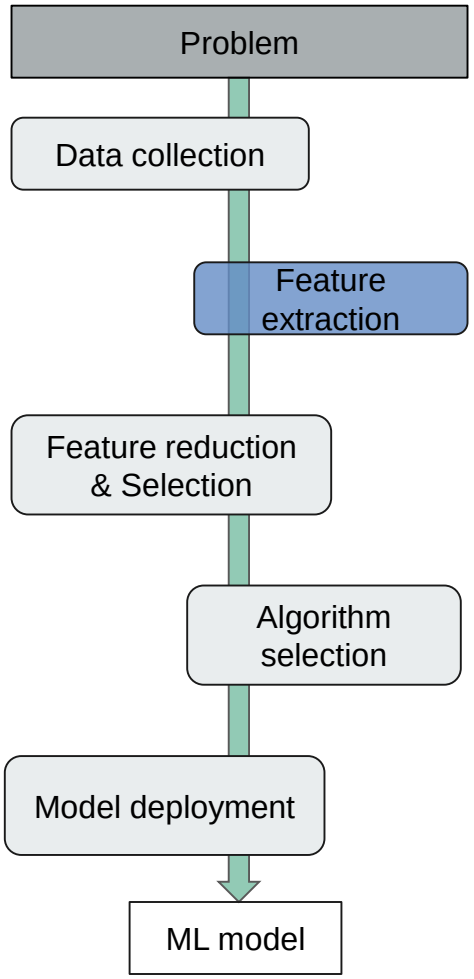


Example of Transport Function graph

TS_1: ***No-error service*** where:

- in Rx, TF3 = *error detect* function and TF0 = *error report* function.
- in Tx, TF1 = *retransmission* function.

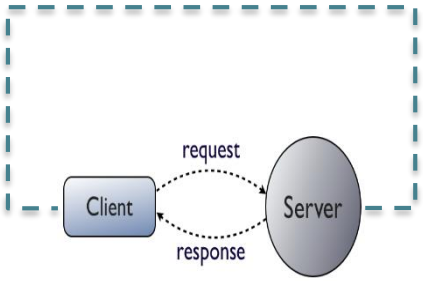
Internet traffic classification



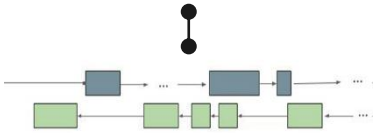
General Machine Learning(ML) procedure

Internet traffic classification

Guarantee the Quality of Service (QoS) by identifying the name of the application given traffic measurements

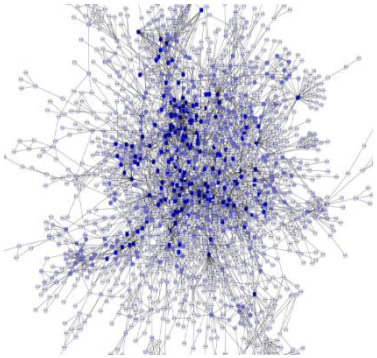


Data transfer



Flows of packets

Thousands of communications, in consequence, **guarantee** the QoS is challenging

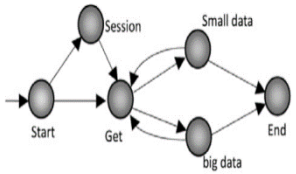


Traffic Classification

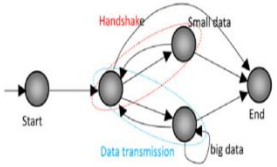
Change the communication settings to improve the QoS

Syntactic structure of some traffic

HTTP

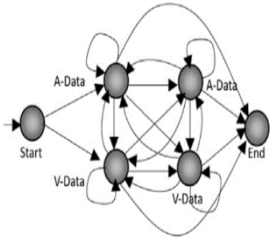


Bit torrent



⋮

Skype



Internet traffic classification

Challenges

Data collection

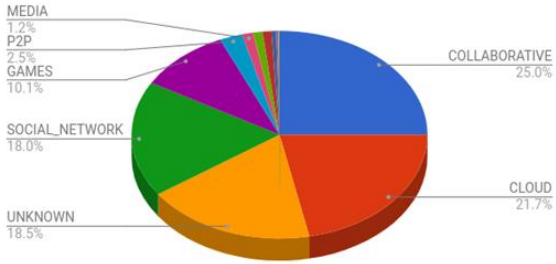
- Real world measurements and tests on the network are difficult to achieve
- Labeling traffic traces can be a tedious task

Feature extraction (FE)

- Encryption and encapsulation can disable classical FE procedures
- It is necessary to propose better descriptors that prevent misclassifications and class imbalance behaviors

Classification

- Presence of class-imbalance in the data, which is the scenario where there are one or more classes with a considerable higher amount of samples than another class(es)
- Class-imbalance data can bias some ML models to learn more from a class than another



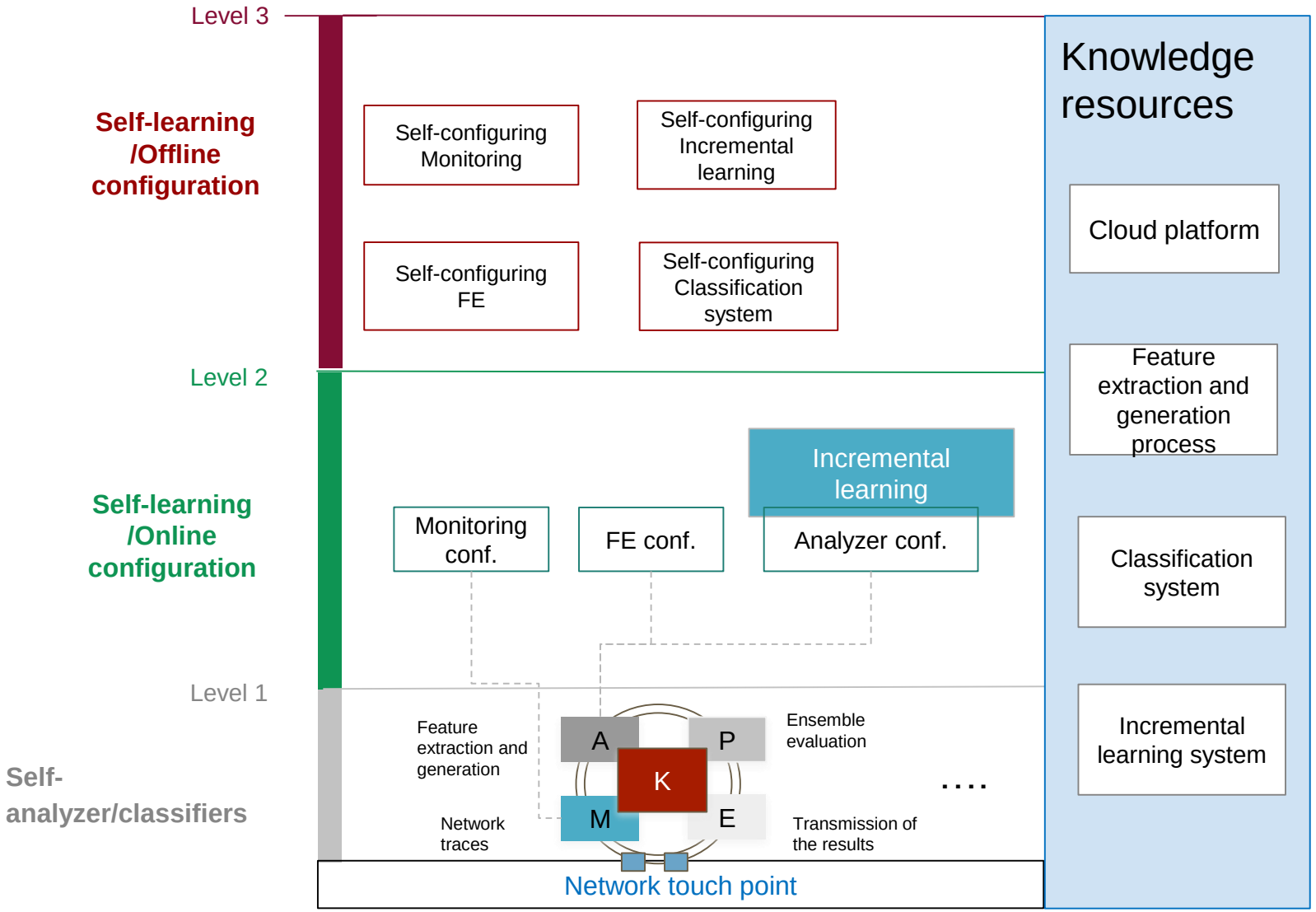
QoS class distribution a Internet traffic dataset

Evolution and dynamism of the data

Validation of the ML solutions

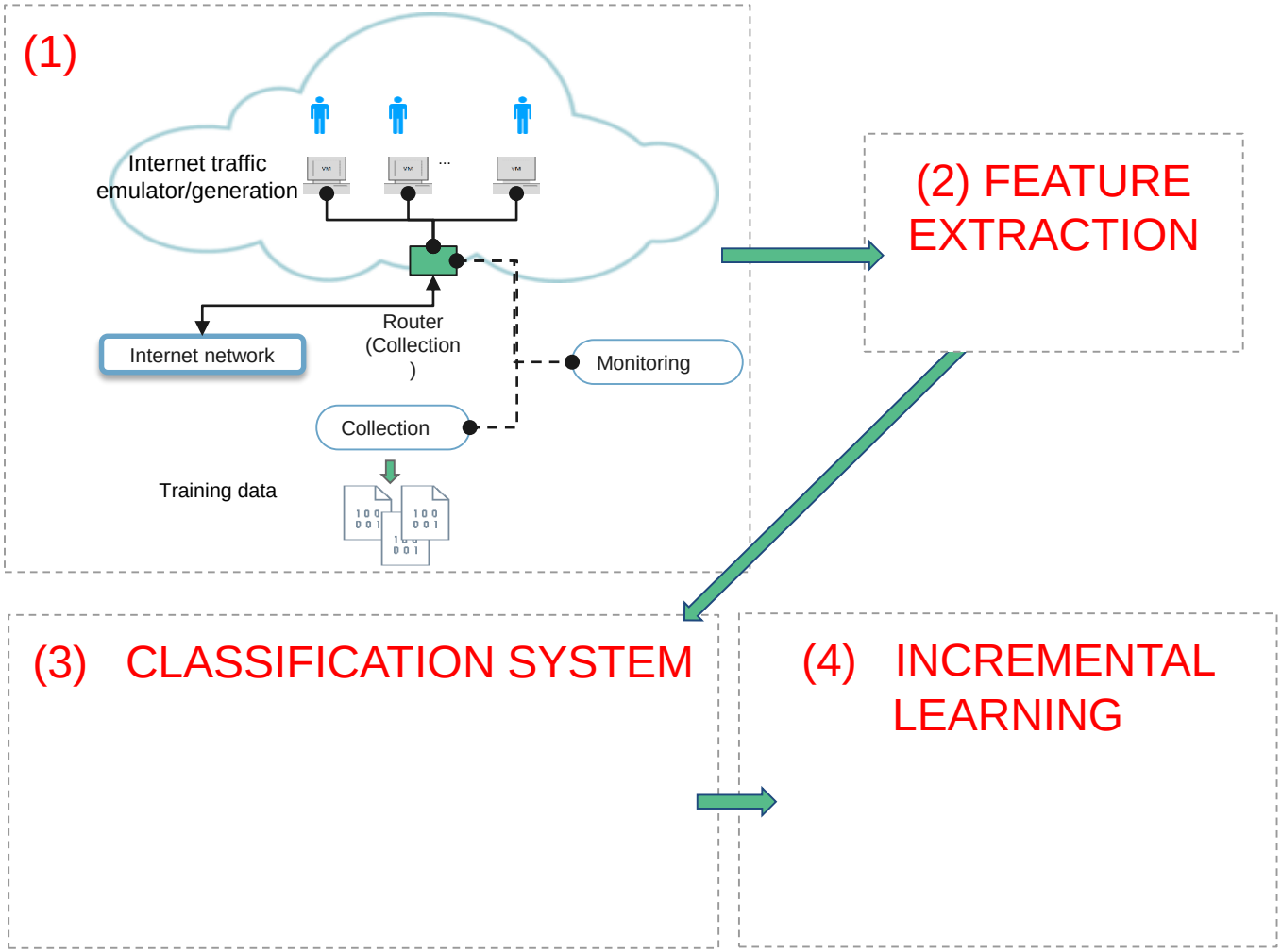
Internet traffic classification

Architectural view

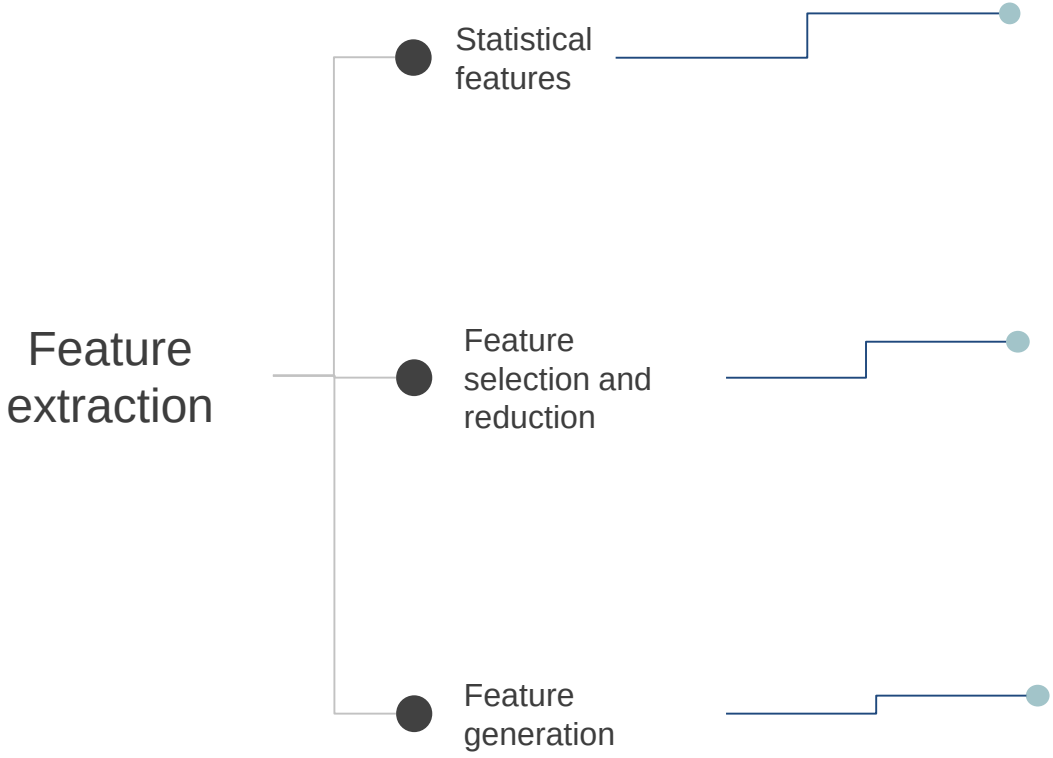


Internet traffic classification

Data collection



Internet traffic classification



- More than 200 statistical features can be computed
- No more than 20 statistical features can be use for encrypted and unencrypted traffic

- Their use is facultative for this particular case

- A statistical based feature extraction approach for the inner-class feature estimation using linear regression

Internet traffic classification

Statistical based feature extraction approach for the inner-class feature estimation using linear regression

Statistical based features

It is the most popular approach

It does not intrude into the packet content

It has a lightweight computation

It shows a high performance for discriminating the applications

$$F_i = \{H_i, P_i, l_i\}$$ Flow of packets

Feature	Description
Packet length	$B = len(p) \ \forall \ p \in P_i$
Inter-arrival time (IAT)	$IAT = t_i - t_{i-1}$

Statistical based features, such as:
Mean
Std
Maximum
Minimum

$$X_i = [x_{1i}, x_{2i}, \dots, x_{1k}]$$

$$F_i' = \{X_i, l_i\}$$

Internet traffic classification

Statistical based feature extraction approach for the inner-class feature estimation using linear regression

Classical approaches: mean

- Moving average

$$\mu_n = \mu_{n-1} + \frac{1}{n}(x_n - \mu_{n-1})$$

- Weighted mean

$$\mu_n = \mu_{n-1} + \frac{w_n}{W_n}(x_n - \mu_{n-1})$$

- Exponential weighted mean

$$\mu_n = \mu_{n-1} + \alpha(x_n - \mu_{n-1})$$

- Logarithmic moving averages

Some remarks

The observations belong to specific **data distributions**.

Online computation can be pruned to errors (**incorrect sampling or noisy-outlier** observations)

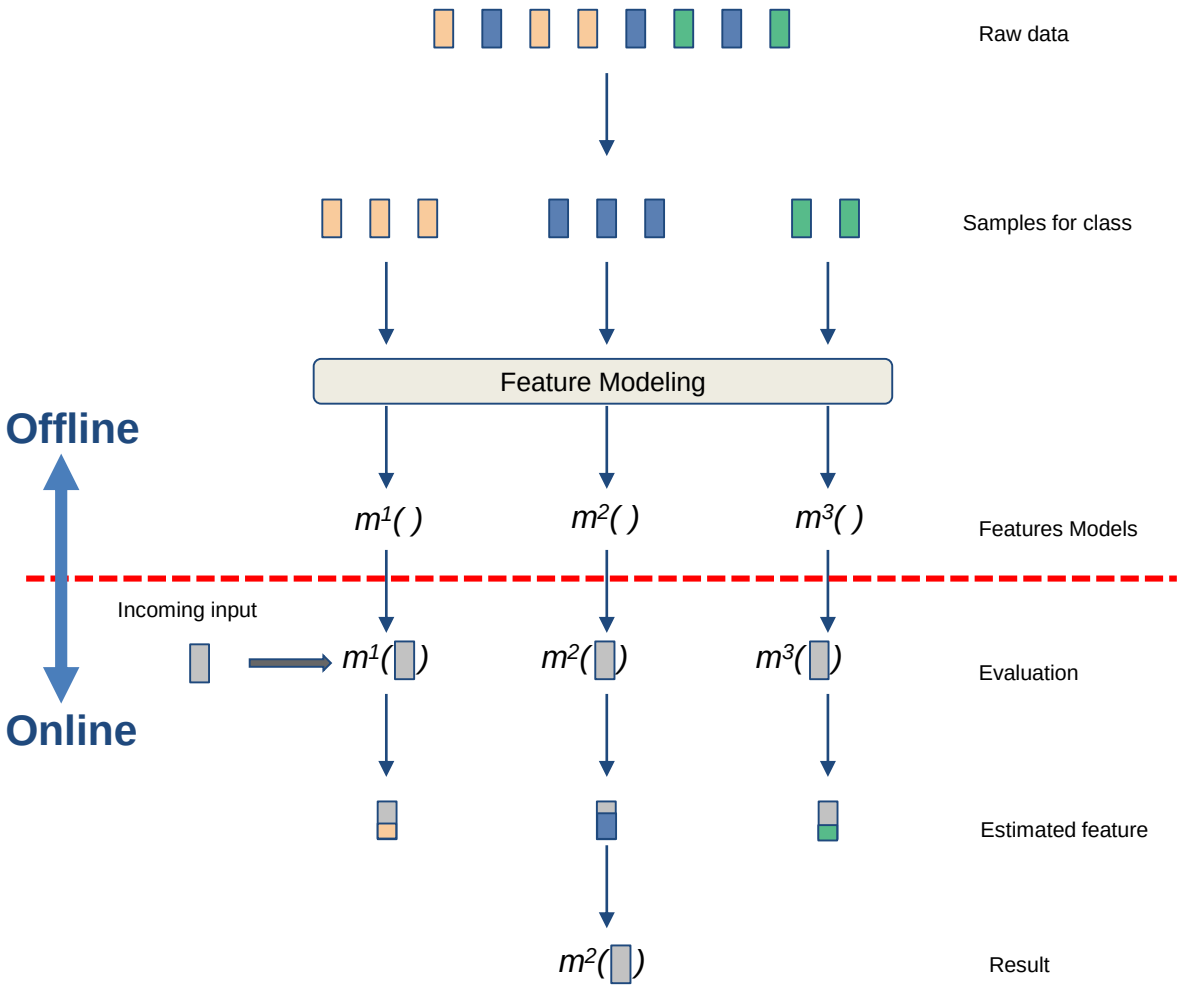
Internet traffic classification

Statistical based feature extraction approach for the inner-class feature estimation using linear regression

Proposal

Assumptions

1. Raw inputs are differentiable from one another
2. The statistical behavior of a variable is different from class to class
3. Statistical features can be modeled for each class separately



Proposal

Estimated features

1. Compute the estimated feature with all the LR models

$a^1=m^1(\square)$ $m^2(\square)$ $m^3(\square)$

1. Compute the distance of each estimated feature against the previous estimation

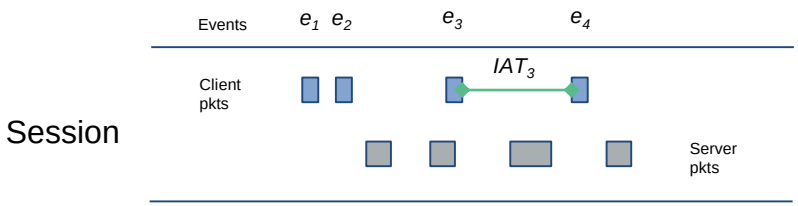
$$d_j^c = [(a_j^c + w) - x_j]^2$$

1. The best approximation is given by the LR model that obtains the lowest distance value

$$a_j = \{a_j^i \in P \mid argmin(d_j^i)\}$$

Internet traffic classification

Experimental evaluation



Event	Moving average	Estimated mean	Moving average	Estimated mean
e_1	μ_1	a_1		
e_2	μ_2	a_2		
. . .	. . .	. . .	. . .	. . .
e_n	μ_n	a_n		

Dataset selected for the experiments

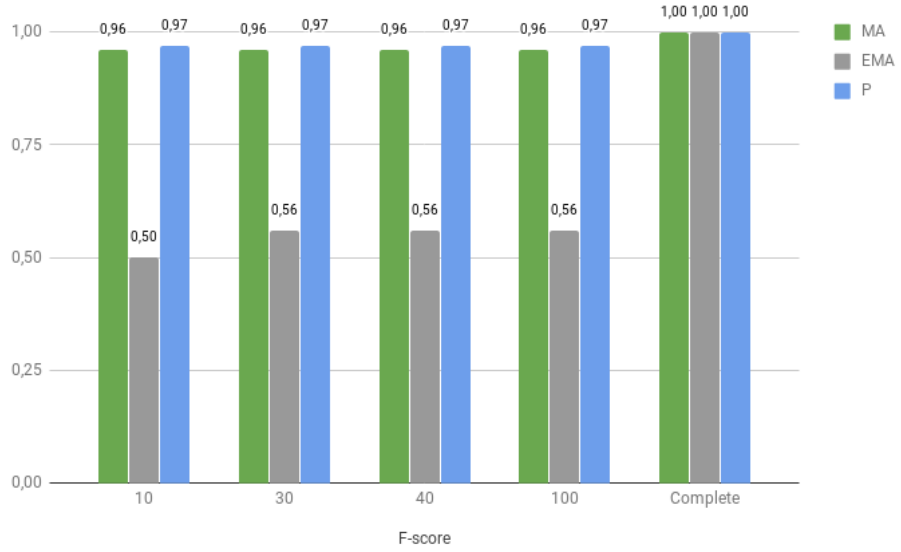
Name	# sessions	# classes
PAM	173429	17

Flow sequence property

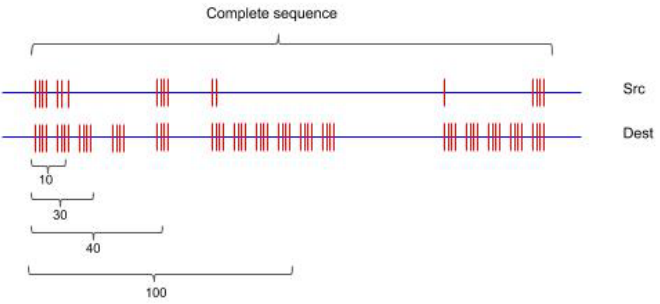
Flow sequence property	Features
Packet length IAT	mean std min max

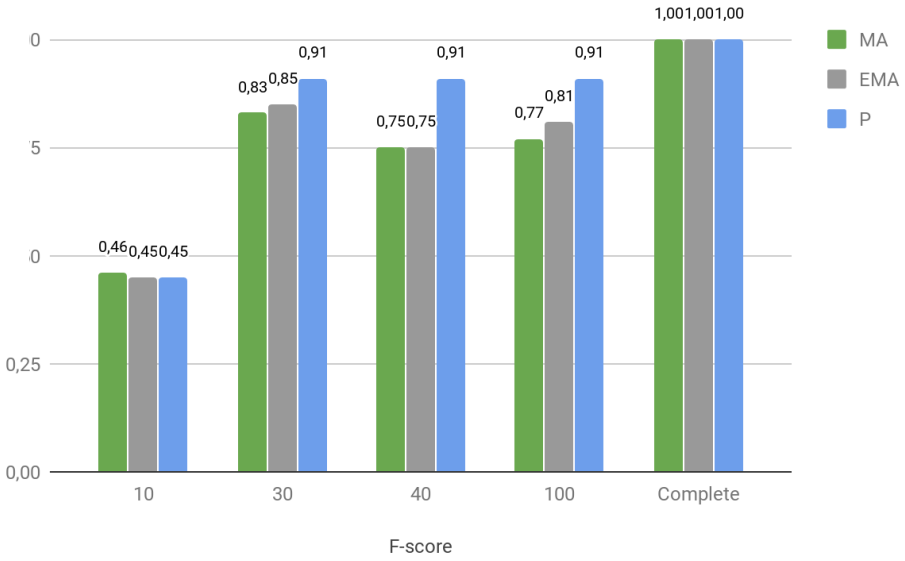
Experimental evaluation

Results: PAM dataset



F-score of the class **Streaming**

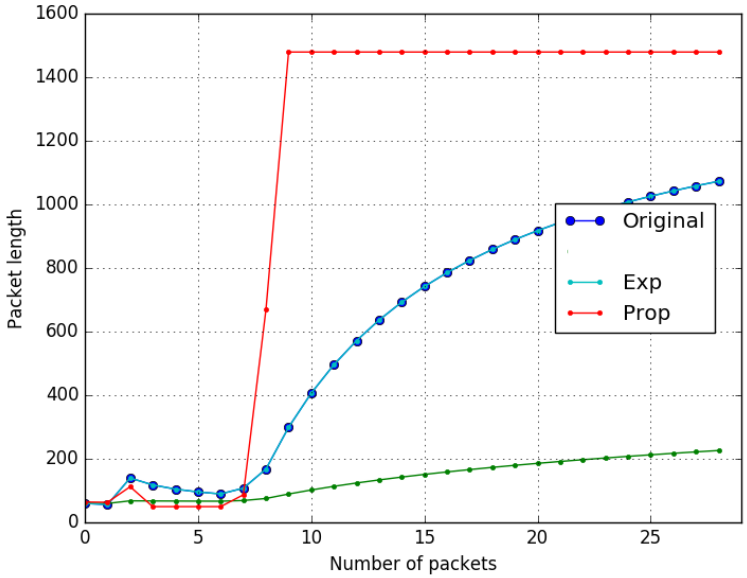




F-score of the class **Web applications**

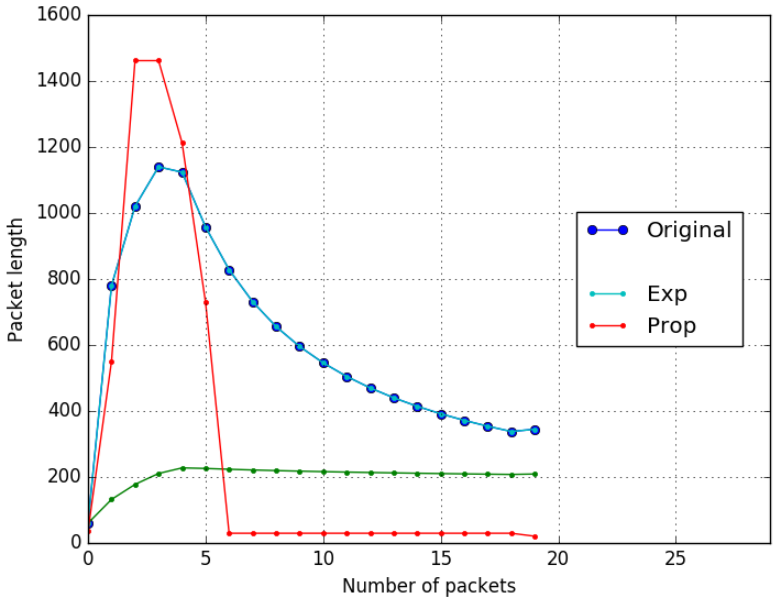
Experimental evaluation

Analysis: PAM dataset



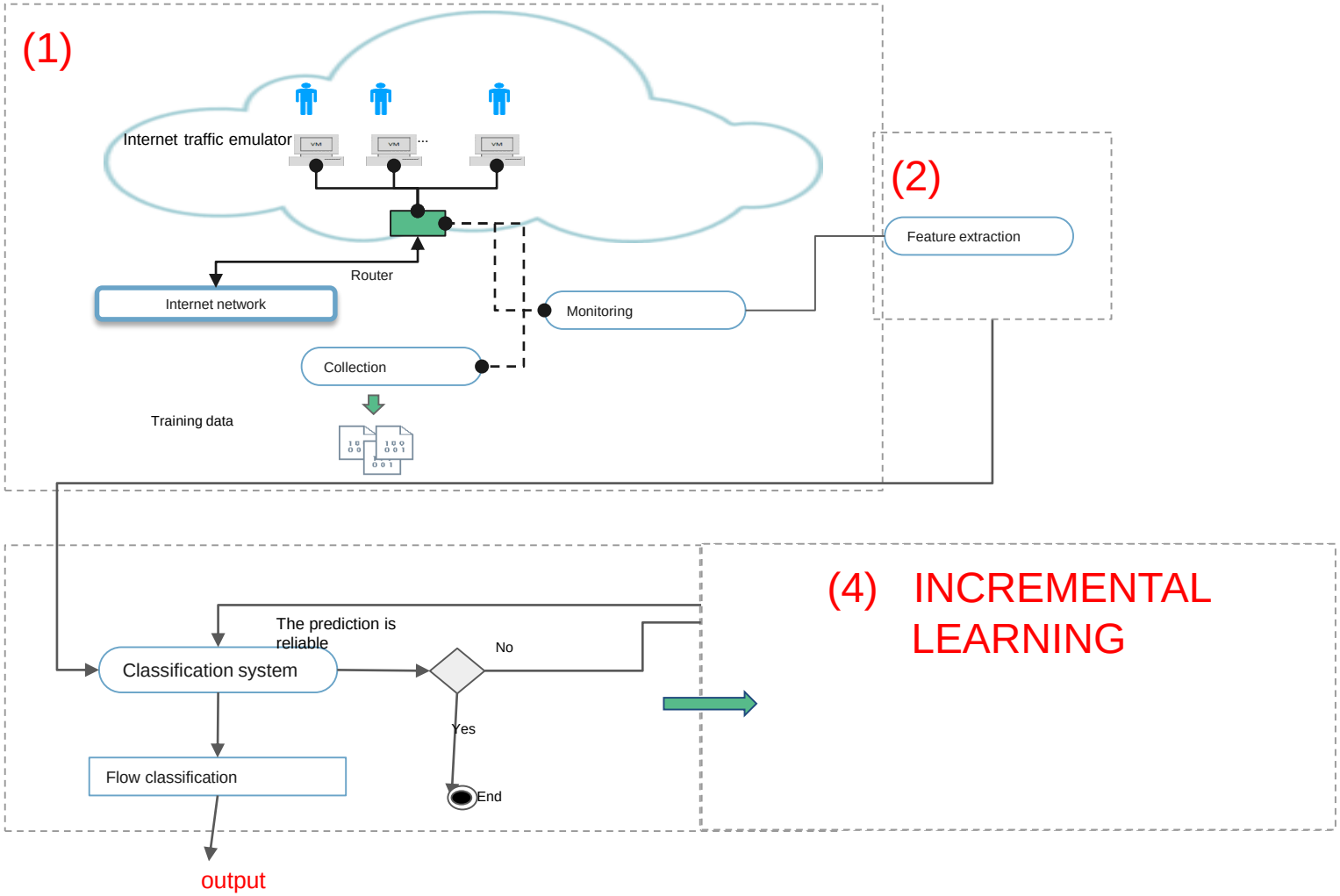
Statistical features for a packet sequence in the **client** with the **VoIP** class

Statistical features for a packet sequence in the **server** with the **VoIP** class

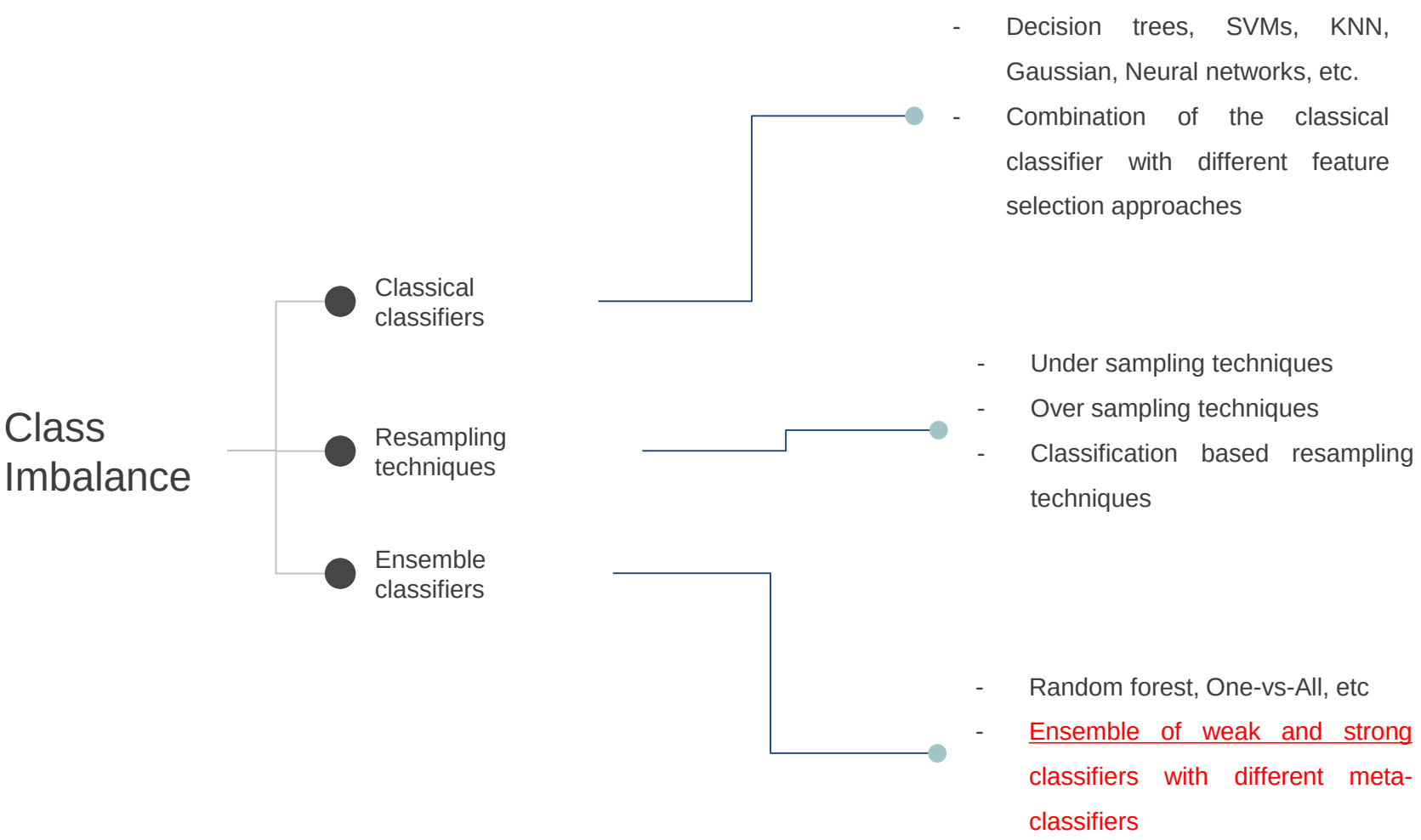


Internet traffic classification

Classification system

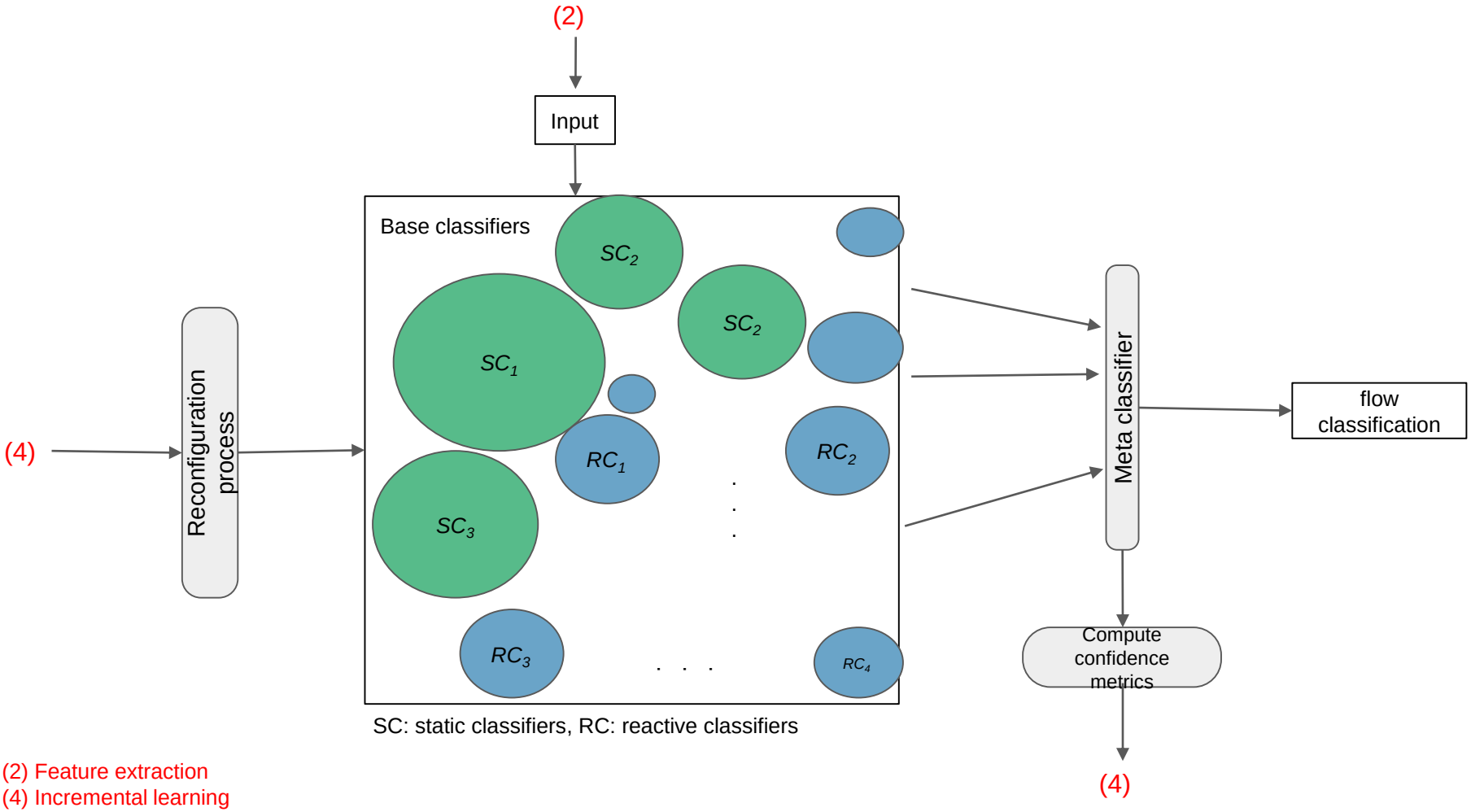


Internet traffic classification



Internet traffic classification

Classification system



SC: static classifiers, RC: reactive classifiers

(2) Feature extraction
(4) Incremental learning

Internet traffic classification

Base classifiers

$$C_i = (name, m_i, t, w_i, F_i, Q_i, s)$$

name = identifier of the classifier

m = model

t = type of classifier, either static or reactive

W = is a vector classification weight that will be used to penalize the classifier by class

F = f-score of the classifier for each class

Q = is a vector that stores the cumulative value of the classifier's quality on predicting an input sample for the class

s = is the class where the classifier is specialized on

Meta classifier

$$y' = G\left(p_1 \times w'_1 \times f'_1, p_2 \times w'_2 \times f'_2, ..., p_k \times w'_k \times f'_3\right)$$

y' = final prediction

G = is the combination function selected

p = probability of membership to the class

w' = weight of each classifier

$Q' = P' \frac{(W' + F')}{2}$ each model for the class predicted by the classifier y'

$$q' = \frac{1}{k} \sum \{Q'\}$$

Reconfiguration process

Adding a reactive classifier

- It will be expert in one class
- The generalization of the ensemble can be lost.
- In order to tackle this problem, the f-score of the classes where the learner is not an expert will be set to zero
- It can be seen that this will implement a mask in the voting system

Pruning the ensemble

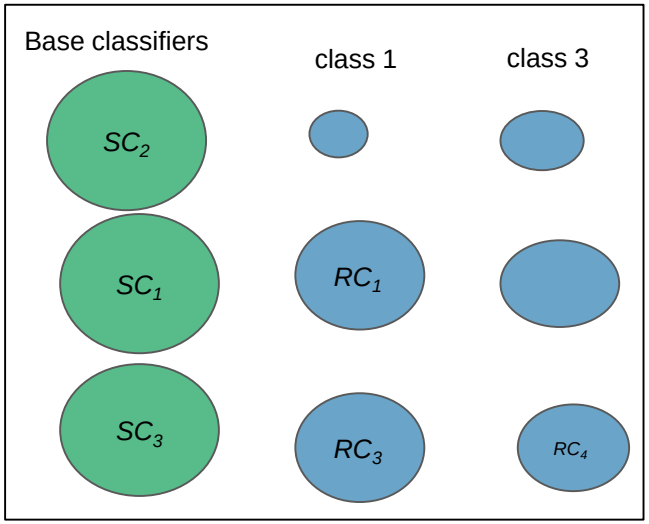
- The amount of RC for class won't be higher than the amount of SC
- Everytime that a new RC is added, if the amount of RC is lower than the SC the ensemble does not change
- However, if this number is equal or higher, the weakest RC is deleted from the ensemble

Penalizing a base classifier

- The class' weights of the base classifiers are updated
- This value is updated when a new RC or SC is added as follows,

$$w_t = w_{t-1} \times \left(1 - \frac{1}{k+c}\right) \mathrm{Q_t}$$

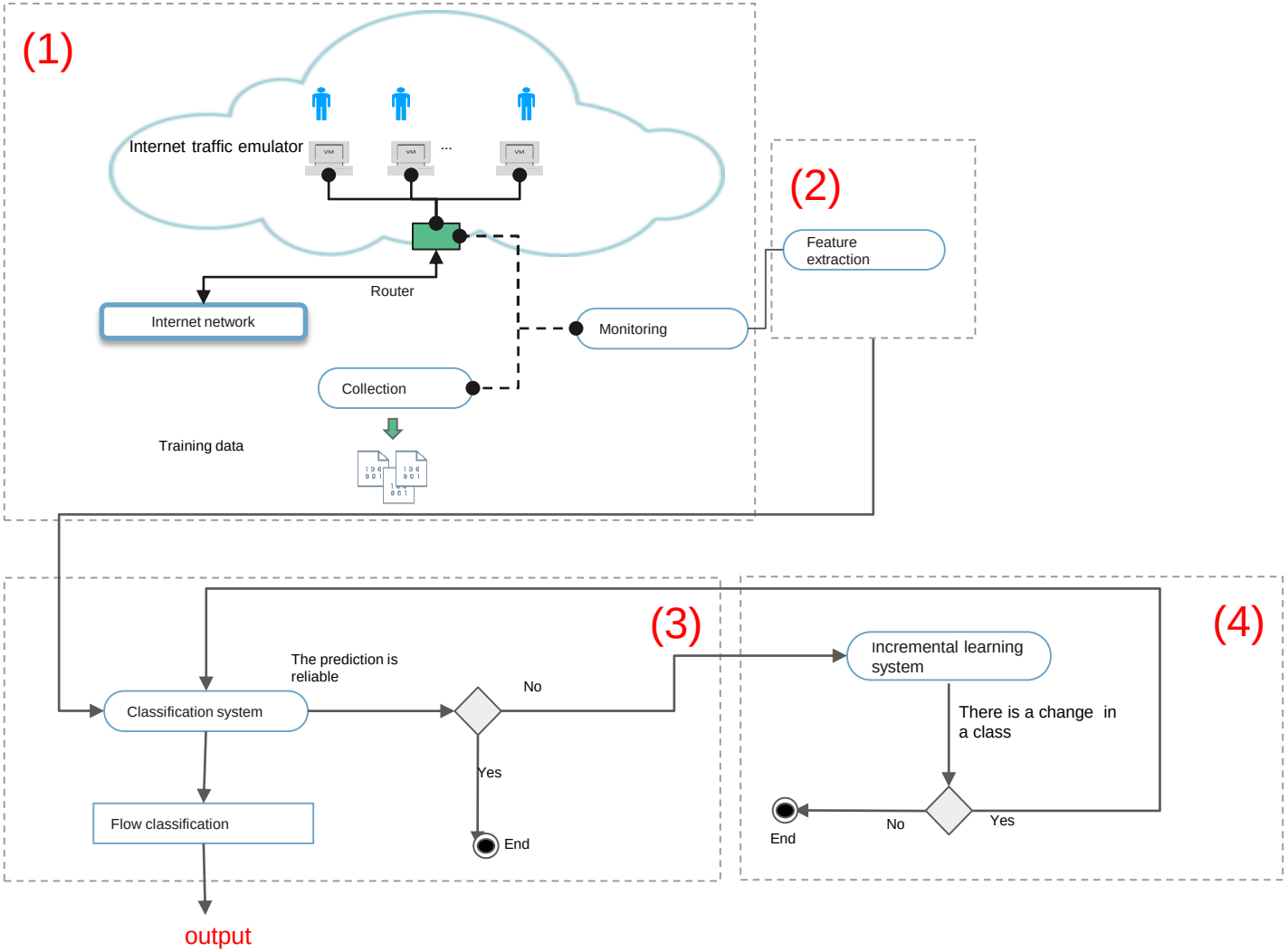
k is the number of classes and *c* the number of classifiers



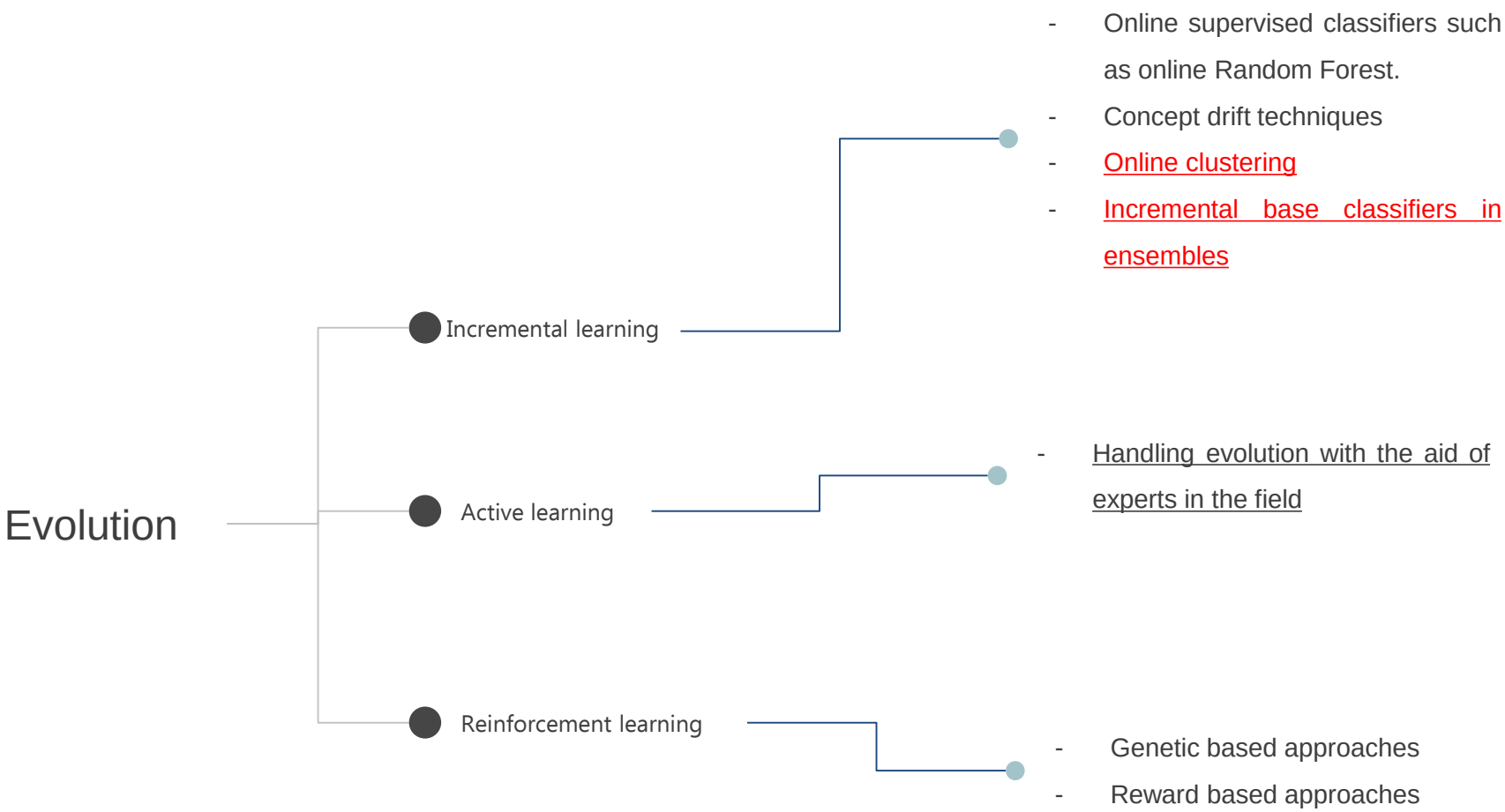
SC: static classifiers, RC: reactive classifiers

Internet traffic classification

Incremental learning



Internet traffic classification

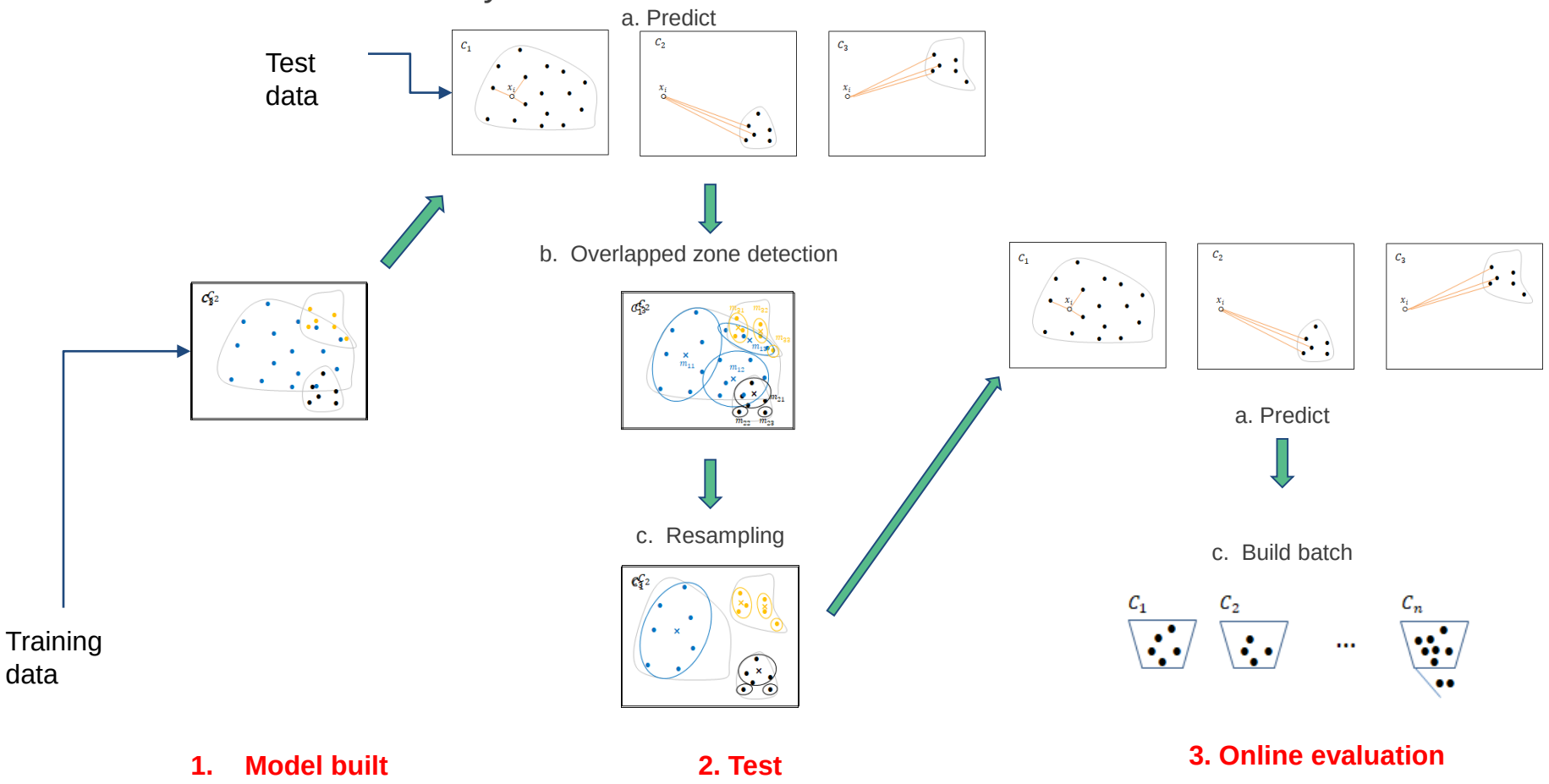


Internet traffic classification

Incremental learning approach

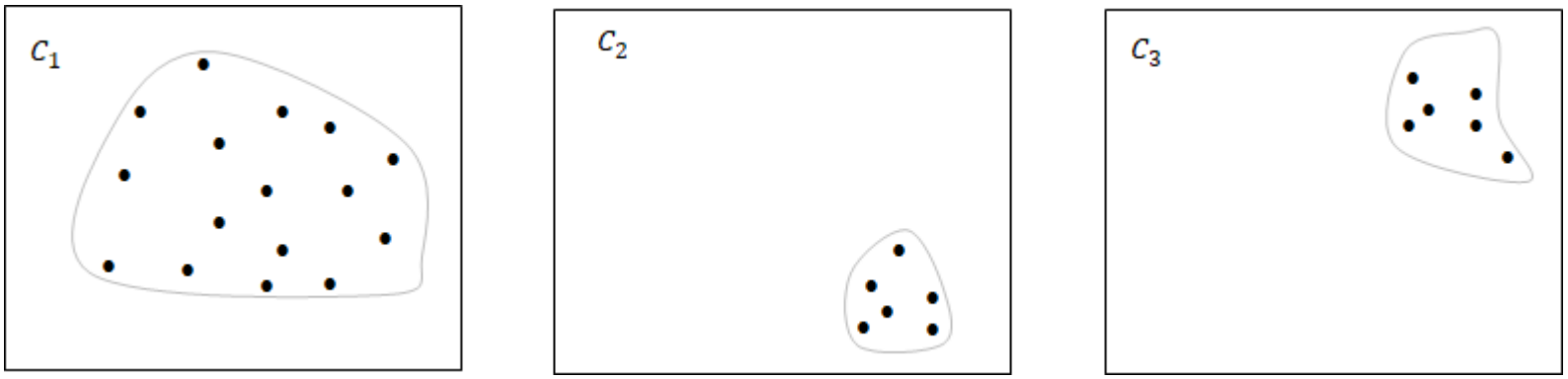
Main properties

- It is an approach based on the principles of one-class classifiers
- One classifier is build by class



Proposal: Incremental learning approach

One-class classifiers



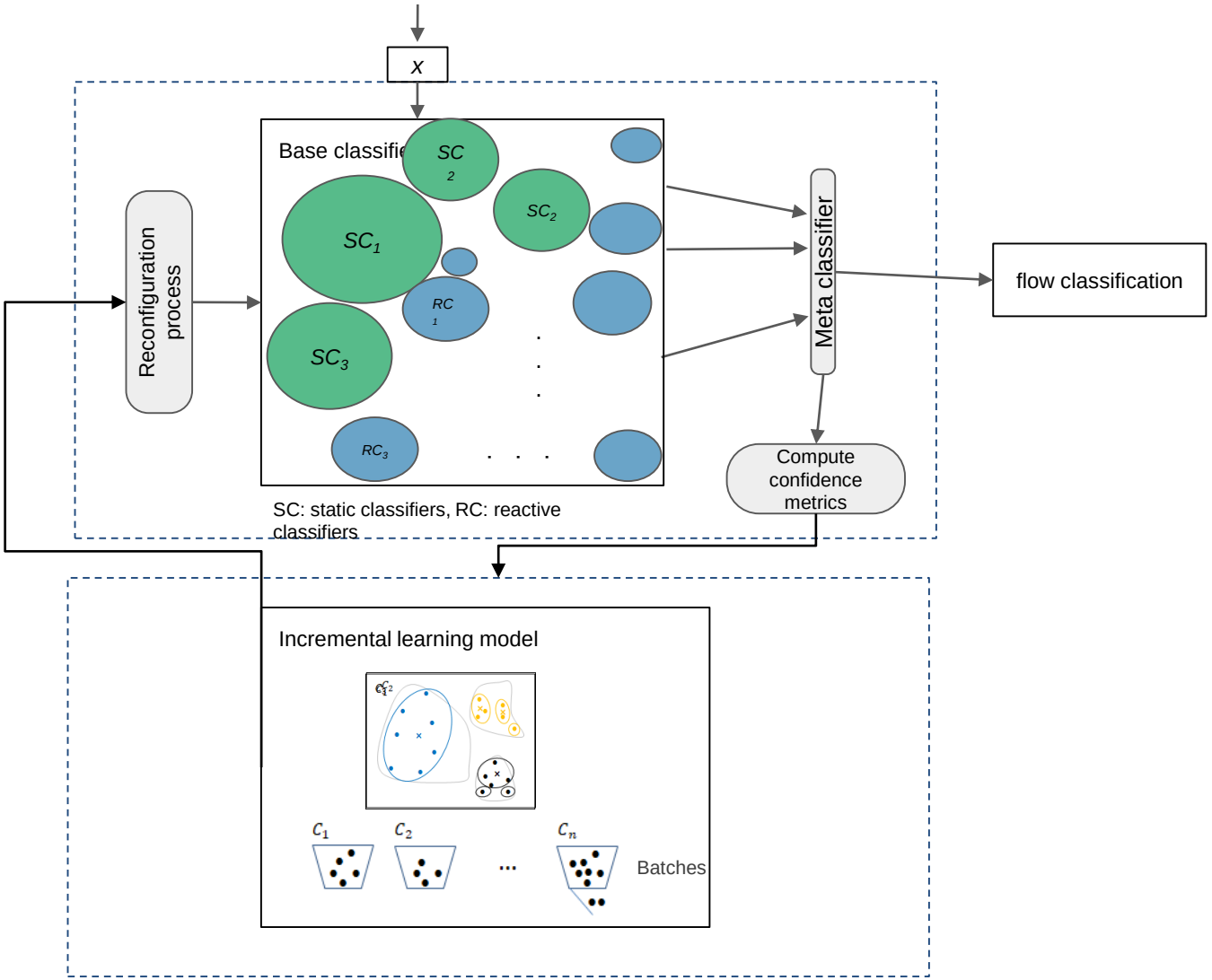
Principles

- The samples of each class are separated
- The classifier is described by a boundary function

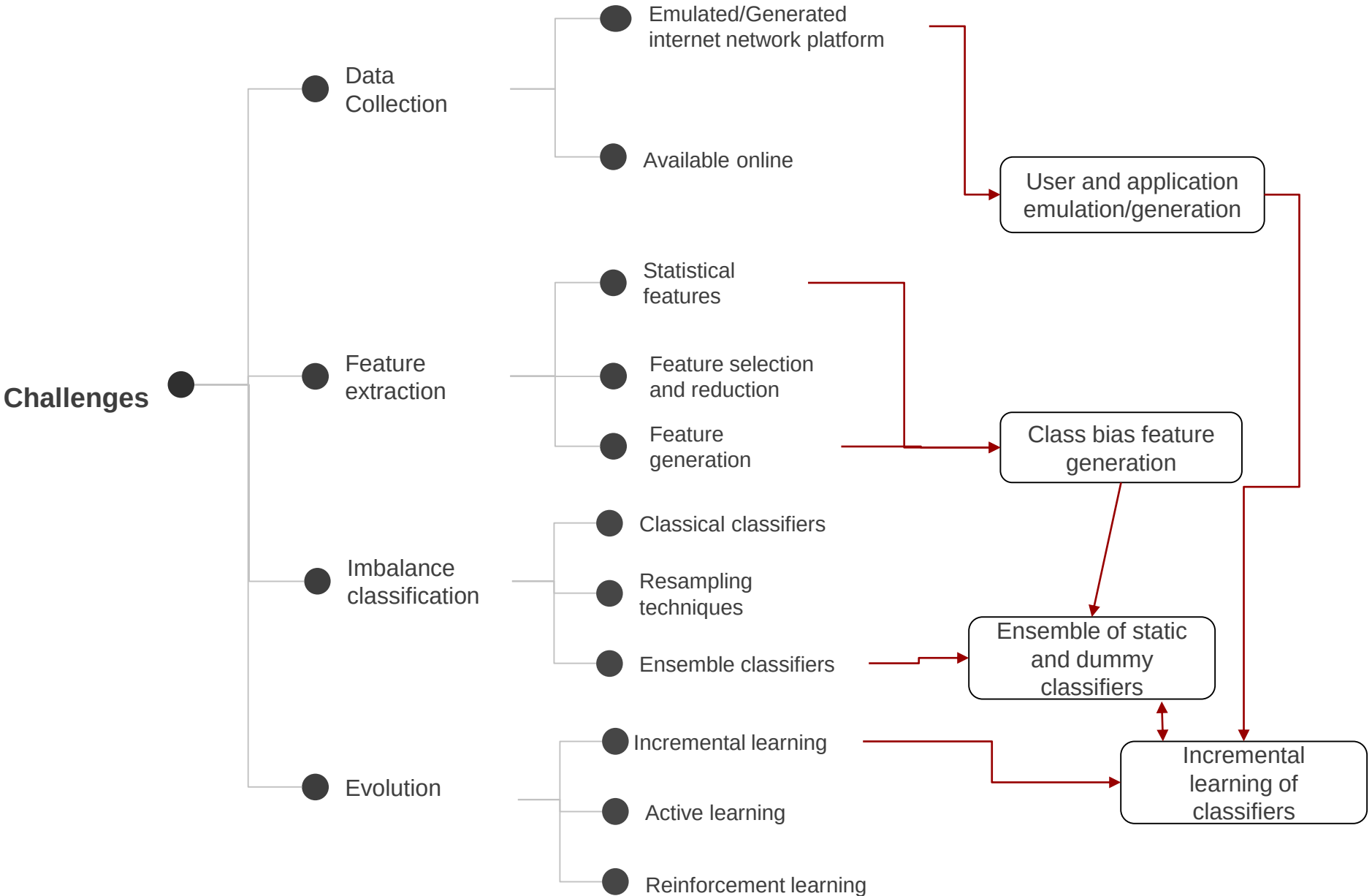
Examples of one class classifiers boundaries functions:

- Density based: Parzen Density estimation and Mixture of Gaussians
- Reconstruction based: Clustering based techniques and auto-encoder neural networks
- Boundary based: One-class Support vector machines, support vector data description, etc.
- Ensemble-based: One-class Clustering-based Ensemble

Internet traffic classification



Internet traffic classification



Quality of Service management in IoT remains a challenge

Summary and Key points to remember:

- A generalization of the vision of network function instantiation (f/V NF);
- Our approach to meet the QoS requirements of IoT applications by, for example, dynamically provisioning Slice at the IoT platform-level;
- IoT platform services / QoS management mechanisms can be packaged in various formats (f/VNF) and deployed dynamically;
- Dynamically deployment allows flexible management needed in environments which vary in time such as IoT, Tactile Internet.
- Traffic classification is a complex problem necessary to consider in the network context

An autonomic network slice provisioning / maintenance cycle

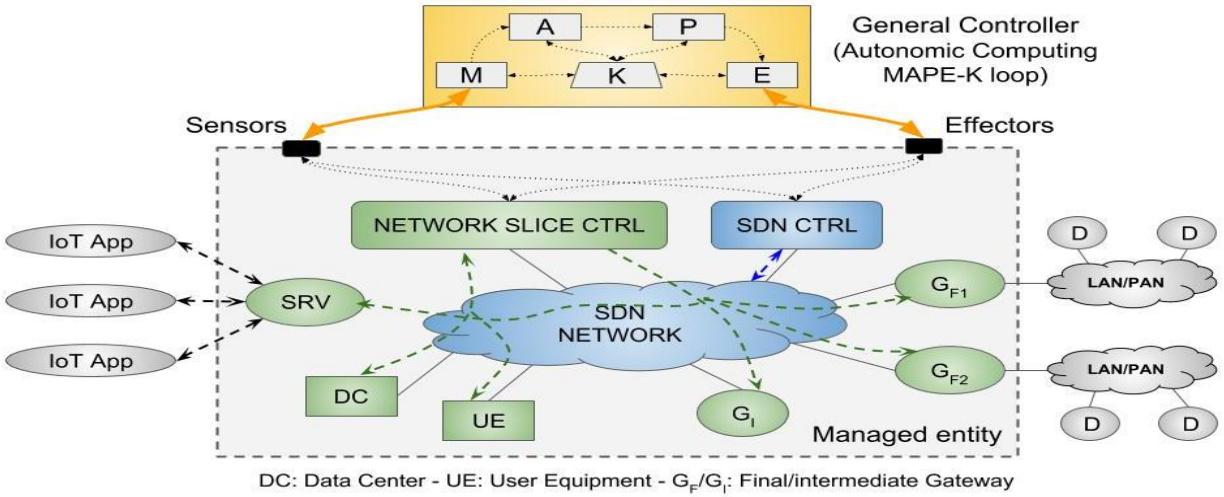
- Add an **abstraction** layer to the considered resources / NOKIA work on NCUs [5]
- **Add tasks** to the current autonomic cycle to consider the **QoE**

[5] S. Sharma, R. Miller and A. Francini, "A Cloud-Native Approach to 5G Network Slicing," in IEEE Communications Magazine, vol. 55, no. 8, pp. 120-127, 2017.

Quality of Service management in IoT remains a challenge

Future works

> Experimental work :
  - A Prototype of the proposed Slice provisioning AC and TFM framework (on going)



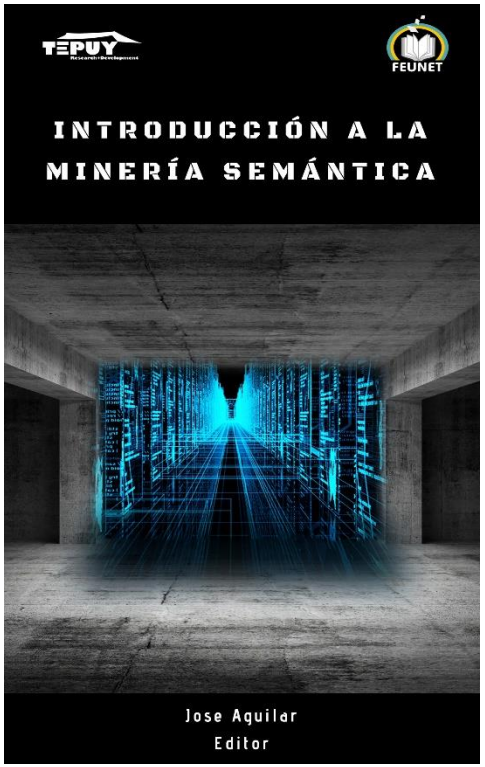
> Theoretical work for the autonomic cycle implementation :
  - Design **structural models** of our architecture (including the deployment environment of QoS-oriented mechanisms) based on the TF graph
  - Continuation with the design of **behavioral models** (descriptive and predictive) of our self-adaptive management system.
  - Development of **noise data processing techniques**
  - Complete the **meta-learning** approach

SOME PAPERS

- F. Pacheco, E. Exposito, J. Aguilar, M. Gineste, C. Budoin, “Towards the deployment of Machine Learning solutions in traffic network classification: A systematic survey”. *IEEE Communications Surveys and Tutorials*, 2018.
- C. Ouedraogo, S. Medjiah, C. Chassot, “Flyweight Network Functions for Network Slicing in IoT”, Proceeding of the *7th IEEE International Conference on Smart Communications in Network Technologies*, 2018.
- J. Aguilar, J. Cordero, O. Buendia, “Specification of the Autonomic Cycles of Learning Analytic Tasks for a Smart Classroom", *Journal of Educational Computing Research*, vol 56 no. 6, pp. 866-891, 2018
- L. Morales, C. Ouedraogo, J. Aguilar, C. Chassot, S. Medjiah, Khalil Drira, “Experimental Comparison of the Diagnostic Capabilities of Classification and Clustering Algorithms for the QoS Management in an Autonomic IoT Platform”, Submit to publication. *Service Oriented Computing and Applications*, Elsevier, 2018.
- E. Bonfoh, S. Medjiah, C. Chassot,“Towards the Virtualization of Transport-level Functions and Protocols”, Coautores: Proceeding of the *7th IEEE International Conference on Smart Communications in Network Technologies*, 2018
- F. Pacheco, E. Exposito, J. Aguilar, M. Gineste, C. Budoin. “A novel statistical based feature extraction approach for the inner-class feature estimation using linear regression”. , Proceeding of the *IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 1005-1012, 2018.
- J. Aguilar, J. Cordero, L. Barba, M Sanchez, P. Valdiviezo, L. Chamba, Learning Analytics Tasks as Services in Smart Classroom’, *Universal Access in the Information Society Journal*, Springer, Vol. 17, No. 4, pp. 693–709, 2018.
- J. Aguilar, M. Jerez, J. Gutiérrez de Mesa, “Context Awareness based on the Network Traffic Information”, Submit to publication. *Journal of Network and Computer Applications*, 2018.
- J. Aguilar, K. Aguilar, C. Jiménez, M. Jerez, "Implementación de Tareas de Analítica de Datos para Mejorar la Calidad de Servicios en las Redes de Comunicaciones", *Publicaciones en Ciencias y Tecnología*, Vol. 11, No. 2, pp. 63-77, 2017.

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

UNIVERSIDAD COMPLUTENSE MADRID



"Insanity is doing the same thing over and over again and expecting different results"
A. Einstein

www.ing.ula.ve/~aguilar

www.ing.ula.ve/~aguilar/actividad-docente/cursos/ConferenceSpain2018.pdf

cemisid
centro de estudios en microelectrónica y sistemas distribuidos

Internet traffic classification

- A novel approach to extract statistical based features is presented
- Procedures to create an ensemble of LR models for each class are defined under defined assumptions
- The workflow reached good performance when the windows of the raw sample is small.
- Inner-class discrimination was improved by our approach