A QoS-Aware Broker for Multi-Provider Serverless Applications

Andrea Graziani
m. 0273395

Supervisor: **Valeria Cardellini**
Tutor: Gabriele Russo Russo

Università degli Studi di Roma "Tor Vergata"
Macroarea di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica

23/05/2022

Research's goal

What is the work goal?

- A **QoS-aware framework** for the orchestration of serverless functions supporting:
  - Multiple FaaS providers
  - Multiple functions implementations.

Why is it necessary to built that classification model?

"The most compelling reason for profiling IoT traffic is to enhance cyber-security. [...] IoT devices are by their nature easier to infiltrate [...] and used to launch large-scale attacks."
ITPAReport

Research's goal

- In other words, if is possible to built a classification model capable to understand IoT devices "*normal*" **traffic pattern**, it will be possible to deploy a network-level security mechanisms to identify attacks analysing traffic patterns.
- According to the authors, that classification model is intended to be used inside a smart cities and campus environments.

How the authors have built their classification model?

- Using a **supervised machine learning** approach, exploiting, form the implementation point of view, a collection of tools and algorithms regarding data analysis called `Weka`.
  All data were been analysed using **Random Forest** algorithm.

What is the first step necessary to start the building of this classification model?

The "*Serverless Computing Paradigm*"

What's meant by "*Serverless Computing Paradigm*"?

An applications service model where:

- Administration tasks (provisioning, monitoring, scaling, etc.) are directly managed by the provider.
- Small-granularity billing pricing model: *pay-as-you-go*.
- Cloud application are abstracted as a group of so-called *serverless functions*.

The "*Serverless Computing Paradigm*"

What is a "*Serverless Function*"?

A **stateless**, **event-driven**, **short-lived**, **self-contained** unit of computation implementing a business functionality.

The "*Serverless Computing Paradigm*"

A serverless function is executed inside a containerized environment called **function instance**.

Any function instance can be in:

- Initialization State.
- Idle State.
- Running State

Warm Pool

The set of all function instances whose state is either **idle** or **running** state.

The "Smart Environment" - Overview - Part 2

The platform **automatically scales** the number of function instances.

When a request comes in, only one of the following events can occur:

- **Warm start**.
- **Cold start.**

We assume that all FaaS providers adopt the auto-scaling technique called *scale-per-request*.

The "Smart Environment" - Overview - Part 2

Concurrency Level

FaaS platforms impose a limitation on the number of function instance runnable at the same time.

Any provider applies these restriction differently:

- Global (Per-Account) Concurrency Model.
  - AWS Lambda.
  - IBM Cloud Functions.
- Local (Per-Function) Concurrency Model.
  - Google Cloud Functions.

The "Smart Environment" - Overview - Part 2

What are the most important limitations of today's FaaS Platforms?

- Cold Starts.
- Too simple scheduling policies (FIFO).
- Security concerns (*side-channel* attacks).
- Branch mispredictions.
- ...

Ok, but which one we tried to solve?

- Quality of service (*QoS*)

The "Smart Environment" - Overview - Part 2

Why is so difficult to guarantee the respect of QoS constraints?

- Lack of performance model.
- Lack of a methodological way to find a suitable "*configuration*" for a serverless application.

What we mean for "*configuration*"?

The "Smart Environment" - Overview - Part 2

Why is so difficult to find a suitable configuration for a serverless application?

Configuration parameters **significantly** affect the **cost** and **response time** of serverless functions.

Fortunately, solutions to that problem **already** exist, **but**:

- They are unaware of the current status of FaaS platforms.
- No support for multiple implementations, or versions, of a same serverless function.
- No support for multiple FaaS platforms scheduling.
- No support for generic workload applications.

Fortunately, solutions to that problem **already** exist, **but**:

- They are unaware of the current status of FaaS platforms.
- No support for multiple implementations, or versions, of a same serverless function.
- No support for multiple FaaS platforms scheduling.
- No support for generic workload applications.

The "Smart Environment" - Architecture - Part 1

- Similarly to many `LPWAN` implementations (`LoRaWAN`, `Sigfox`), proposed architecture can be splitted into:

  Front-end which contains the *router* and the *IoT/non-IoT devices*.
  - Both *wireless* and *wired* interfaces are used.

  Back-end represented by the *cloud*.
  - IoT devices rely on cloud back-end services for **data storage**, **backup**, **firmware updates**, **media streaming** [**mazhar2020characterizing**] ... and **remote access and integration using RESTful Web API**.

The "Smart Environment" - Architecture - Part 2

- In order to preserve battery life, expensive and complex task are **offloaded** to the back-end system, that is to the cloud.
  In other words, cloud resources are exploited through so-called **cyber-foraging techniques** in order to overcome the very strictly constrains of IoT devices. [**TheSeminalRoleEdgeNativeApplication**
- Since the cloud represents the main execution site for tasks and services, proposed architecture is intended for **cloud-native** IoT applications and services [**TheSeminalRoleEdgeNativeApplications**

The "Smart Environment" - Architecture - Part 3

Proposed architecture is based on a **star network topology**.

The use of a star network topology allows us to:

- **Preserve battery life** of IoT devices because they **do not have to forward** other nodes data; in other words, *any IoT device receives, or transmits, only its own data* [**LoRaWAN**].
- **Decrease the complexity** of the network [**LoRaWAN**] making infrastructure deployment cost low.

IoT Traffic - Part 1

"[...] if we consider only the load imposed by the IoT devices, then there is a dramatic reduction in the peak load (1 Mbps) and average loads (66 Kbps), [...], implying that traffic generated by IoT devices is small compared to traditional non-IoT traffic."
ITPAReport

"the traffic pattern of one IoT device [...] a pattern of active/sleep communication emerges. [...] IoT active time [...] decays rapidly initially (only 5% of sessions last longer than 5 seconds), with the maximum active time being 250 seconds in our trace. This shows that IoT activities are short-lived in general."
ITPAReport

IoT Traffic - Part 2

- Since many IoT devices are battery powered, in order to maximize energy efficiency, results provided by researchers state that the power management approach adopted by IoT devices is based on **periodic sleep**, during which radio transceiver are turned off.

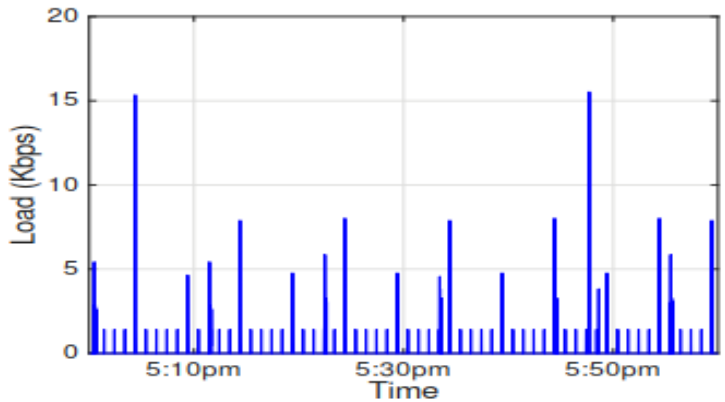
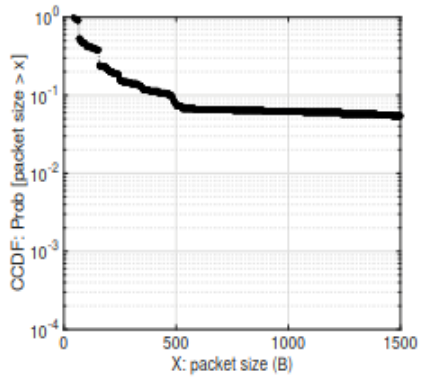


Figure: Load of LiFX light bulb device.

IoT Traffic - Part 3

- A very interesting observation regards packet size. Experimental results state that only the 10% of packets are larger than 500 Bytes.



"More than 75% of IoT sessions transfer less than 1 *KB, and only fewer than 1% of the sessions exchange more than 10 KB, suggesting that the majority of IoT devices generate only a small burst of traffic."*
ITPAReport

The "Smart Environment" - Wireless Technologies - Part 1

- Researchers state that all wireless IoT devices support `IEEE 802.11` as wireless technologies.
- Researchers did *not* specify which **version** of IEEE 802.11 standard has been effectively used. Moreover, we don't know with which **frequencies** data has been transmitted.
  - However we can learn more from the vendor's specifications regarding the TP Link Archer C7, which supports following protocols:
    - `IEEE 802.11ac/n/a` at 5 GHz
    - `IEEE 802.11n/b/g` at 2.4 GHz

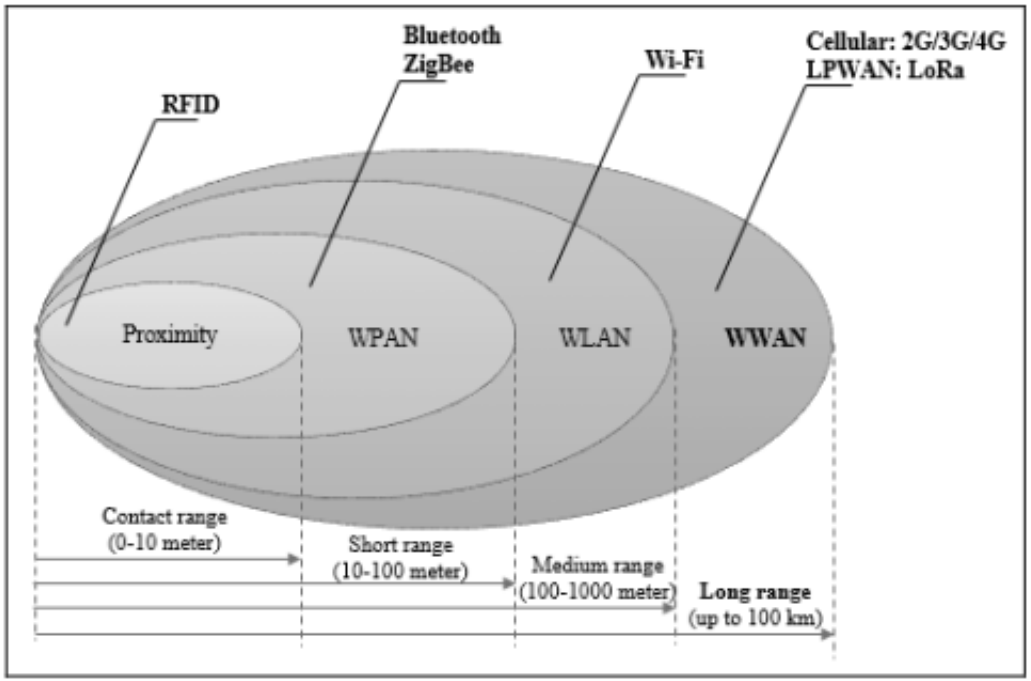
The "Smart Environment" - Wireless Technologies - Part 2

`IEEE 802.11` wireless technologies can **not** be fully optimized for IoT business models and devices used in **smart cities** and **campuses**.

- Despite high reliability, low latency, and high transfer rates (using `802.11ac` 5 GHz we can achieve 1300 Mbps), due to their **inherent complexity** and **energy consumption** are generally not suitable for all IoT nodes [**IOTCITY**].
  - All results regarding IoT traffic suggest that IoT devices requires **low power** and **less data-rate**. In this context, any `LPWAN` solution, like `LoRaWAN`, can be a better choice [**IOTCITY**].
- These technologies provide a **short/medium coverage** with **100-to-1000** meters range [**LoRaWAN**]. Provided coverage range can be not enough to fulfil all use cases inside a smart-

The "Smart Environment" - Wireless Technologies - Part 3

- Utilizing sub-1 GHz bands is possible to provide **better propagation characteristics** in outdoor scenarios. **Low frequencies signal are less affected by obstacles presence.**



The "Smart Environment" - Wireless Technologies - Part 3

Adopted wireless technologies are suitable only for *unconstrained* IoT devices which requiring **short-medium** range scenarios.
Seem that experimental setup not cover all smart-city and campus use cases...

The Most Dominant Application Layer Protocols

Experimental results regarding the **application layer protocol** (inferred using the destination port numbers) that IoT devices use to communicate, show that:

- HTTPS (TCP port 443) is the dominant protocol used by the IoT devices since it represents over the 55% of total IoT traffic.
- HTTP (TCP port 80) represent the second most dominant application layer protocol constituting the 11% of total traffic.
- SSDP (UDP port 1900) is the next most dominant application layer protocol representing the 8% of traffic.
- RTMP (TCP port 1935) represent the fourth most dominant protocol representing the 7% of traffic
- DNS (UDP port 53) represents less than 5/4% of total traffic.
- NTP (UDP port 123) constitutes less than 2/3% of IoT traf-

The role of HTTP - Part 1

Why HTTPS and HTTP are the most used protocols?

- A very important aspect of an urban, or campus, IoT infrastructure is the **necessity to make data collected by the urban IoT devices easily accessible** by both authorities and citizens [**IOTCITY**].
- Generally, IoT devices do not conform to a unified *application programming interface* (API) so that a developer has to learn various protocols and APIs of each IoT device.
  Therefore, in order to develop IoT applications, in order to integrate a lot of heterogeneous communication protocols and system software, is still complex and costly.
  - In order to achieve this objective, IoT devices adopt a very well known web-based paradigm called **Representational State Transfer** (**ReST**), which plays a very important role into **Web**

The role of HTTP - Part 2

- Exploiting REST paradigm, HTTP and HTTPS are used very frequently because they facilitate both the **integration of IoT devices** with existing services currently available on the Web and the **Web applications development** [**IOTCITY**][**WOT**].
  - HTTP and HTTPS offer a **direct access** for users to IoT devices data and services, without the need for installing additional software [**IOTCITY**].
    In fact, using a Web browser (or any HTTP library in the case of a software client) client are able to to directly extract, save and share smart things data and services [**WOT**].

Not only HTTP

- Not all IoT devices support HTTPS/HTTP or support RESTful paradigm.
- Results shows that, regarding remaining IoT traffic, several IoT device use an own **application-specific** protocol.

Device	Belkin switch	Blipcare BP meter	HP printer	Insteon camera	LiFX bulb
port number	TCP 3478	TCP 8777	TCP 5222	UDP 10001	TCP 56700

Device	NEST Protect	Netatmo weather	TPLink camera	Triby speaker	Withings camera
port number	TCP 11095	TCP 25050	TCP 50443	TCP 5228	TCP 1935

The disadvantages of HTTP

- The **verbosity** and **complexity** of native HTTPS/HTTP make them **unsuitable** for constrained IoT devices [**IOTCITY**].
  - In fact, the **human-readable format** of HTTP, which has been one of the reasons of its success in traditional networks, turns out to be a limiting factor due to the large amount of heavily correlated (and, hence, **redundant**) data [**IOTCITY**].
- HTTPS/HTTP rely upon the TCP transport protocol that, however, does not scale well on constrained devices, yielding poor performance for small data flows in lossy environments [**IOTCITY**].
- In a Smart City environment, can be **not** possible to install a full stack of HTTP into resource-constrained devices or sensors.

Seem that experimental setup not cover all smart-city and campus use cases...

Security Problems Due To Unencrypted Traffic - Part 1

- According to **ITPAReport**, about 45% of IoT traffic is **not** sent over HTTPS to the servers.
  - Since the traffic transmitted using other protocols are typically not encrypted, **ITPAReport**'s results indicate that a sizeable fraction of IoT traffic is **not** being securely transported over the Internet.
  - The use of unencrypted protocols can leak sensitive information about users [**mazhar2020characterizing**].

Security Problems Due To Unencrypted Traffic - Part 2

Why IoT devices transmit unencrypted data?

There may be various reasons according to which data are transmitted unencrypted:

- Due to **limitations and constrains** in the IoT device itself.
- As noted by **mazhar2020characterizing**, IoT devices vendors may be hesitant to move to HTTPS if their products use any third-party resources that are HTTP-only. [**mazhar2020characterizing**
- Bad design.

Machine learning - Part 1

Since **ITPAReport** adopted a **supervised machine learning algorithms** to build their classification model, is necessary to generate a **data-set** in order to provide an appropriate input during the **learning phase**.

- **ITPAReport** collected traffic over 3 weeks generated from the "*Smart Environment*".
  - 2 weeks of data was used for *training* and *validation*.
  - last week was used for *test*.
- Is very important to precise that collected data are **time series**, where each instance, indexed by time, contains several **attributes** (or **features**) like *sleep time*, *active time*, *average packet size* and so on.
- Clearly, every instance contains a **label** identifying the IoT device, which is necessary during supervised learning.

An Unbalanced Data-Set - Part 1

The performance and the interpretation of a IoT device classification model **depend heavily on the data** on which it was **trained**.

- Scientific literature showed that classification model, which are trained on *imbalanced datasets*, are highly susceptible to producing inaccurate results.

- Could **ITPAReport**'s dataset be unbalanced?
- Could **ITPAReport**'s classification model be unsuitable to correctly classify IoT devices in a real smart-city and campus scenario?

An Unbalanced Data-Set - Part 2

- Generally smart city and campus services are based on a **very heterogeneous set of IoT devices**, generating **very different types of data** that have to be delivered through **suitable communication technologies**.
  - For instance, possible IoT use cases can be:
    - Structural Health of Buildings.
    - Waste Management.
    - Air Quality.
    - Traffic Congestion.
    - Noise Monitoring.
    - City Energy Consumption.
    - Smart Parking.
    - Smart Lighting.

- Proposed "*Smart environment*" simulates too few IoT use cases relating to a smart-city or campus.
- Proposed environment is more suitable for a *smart-home*

An Unbalanced Data-Set - Part 3

- Too few use cases. Only short/medium range use cases.
- It includes **only** *unconstrained protocol stack*, that is protocols that are currently the de-facto standards for Internet communications and are commonly used by regular Internet hosts (`HTTP/TCP/IPv4`). These protocol are suitable only for unconstrained IoT devices [**IOTCITY**].
  - In fact there is a prevalence of `HTTPS/HTTP` application layer protocol (66% of total IoT traffic according to **ITPAReport**) and of the `TCP` transport layer protocol (representing, more or less, the 85% of total transmitted packets according to **ITPAReport**'s results).
- It does **not** include any *constrained protocol stack*, the low-complexity counterparts of the de-facto standards for Internet, i.e., **Constrained Application Protocol** (`CoAP`), `UDP`, and `6LoWPAN`, which are suitable even for very constrained devices [**IOTCITY**].

Validation technique

"Our cross-validation method randomly splits the dataset into training (90% of total instances) and validation (10% of total instances) sets. This cross-validation is repeated 10 times. The results are then averaged to produce a single performance metric."
ITPAReport

- Since their datasets contains **time-series data**, in order to take into account the time-sensitiveness of data, is required a validation technique, that is a *technique which defines a specific way to split available data in train, validation and test sets*, capable **to preserve the temporal order of data**, preventing for example that the testing set contains data antecedent to the training set.
- Traditional methods of validation, like *10-fold cross-validation*