**317959**

# Mobile Opportunistic Traffic Offloading

# D5.3 – Evaluation of offloading strategies based on experimentation

**(public)**

SEVENTH FRAMEWORK PROGRAMME

| | |
|---|---|
| Grant Agreement No. | 317959 |
| Project acronym | *MOTO* |
| Project title | Mobile Opportunistic Traffic Offloading |
| Deliverable number | D5.3 |
| Deliverable name | Evaluation of offloading strategies based on experimentation |
| Version | V 1.0 |
| Work package | WP 5 – Experimental validation |
| Lead beneficiary | FON |
| Authors | David Valerdi, Alaitz Garcia, Ivan García (FON), Filippo Rebecchi, Farid Benbadis (TCS), Patricia Ortiz, YuanYue Ding, Ivan Prada (INNO), Engin Zeydan (AVEA, Mehdi Bezahaf (UMPC), Andrea Passarella, Giovanni Mainetto (CNR) |
| Nature | R – Report |
| Dissemination level | PU – Public |
| Delivery date | 30/10/2015 (M36) |

## Executive Summary

This document corresponds to the deliverable D5.3 "Evaluation of offloading strategies based on experimentation", part of task 5.3 of FP7-MOTO project. The main purpose of the task is to assess selected aspects of the protocols defined in WP3 and WP4 by testing a MOTO prototype under realistic scenarios.

The deliverable describes the main features implemented in the prototype. It defines the testing plan by splitting the tests into functional and performance ones. The testing plan also includes the description of setups used during the testing, having two main setups: multi-operator and "only Wi-Fi". Testing was performed along three test fests held in April (Istanbul), July (Istanbul) and September (Getxo, Spain) 2015. Prototype was evolved implementing main features (incl. security framework) until a version that allowed executing performance tests and obtaining meaningful results. Analysis of results confirms expectations like DROID offloading algorithm providing a better performance across scenarios. It also provides interesting findings. For instance, MOTO algorithms showed better battery consumption figures than having retrieved the content directing from the WAN. Testing also confirmed that security features, in the way that were implemented, show a negligible impact on the performance.

## Acronyms

**Table 1: Acronyms**

| Acronym | Meaning |
|---|---|
| AP | Access Point |
| API | Application Programming Interface |
| D2D | Device to Device (comms.) |
| DT | Delay Tolerance |
| GPS | Global Positioning System |
| IMSI | International Mobile Subscriber Identity |
| LTE | Long Term Evolution |
| MME | Mobility Management Entity |
| MSISDN | Mobile Subscriber Integrated Services Digital Network Number |
| PZ | Panic Zone |
| QoS | Quality of Service |
| SLA | Service Level Agreement |
| T2T | Terminal to Terminal |
| UE | User Equipment |
| D2D | Device to Device (comms.) |
| DT | Delay Tolerance |
| WAN | Wide Area Network |

# Table of Contents

# List of Figures

# List of Tables

# 1 OVERVIEW

## 1.1 Goals

Testing plan described in this document aims to fulfill the following high level goals:

- Ensure the proper implementation and functioning of the prototype as per design by the execution of a group of functional tests.
- Assess the performance of MOTO framework by the execution of a group of performance tests. Performance assessment is based on metrics like offloading gain, delay from first injection, panic zone ratio, end-user terminal battery consumption, etc. The performance testing is covering the following scenarios:
    - Testing scenarios that allow the assessment of offloading gains across the different offloading algorithms and parameterization and under different scenarios.
    - Testing scenarios that allow the assessment of the QoS (e.g. based on delay from first injection) across the different offloading algorithms and parameterization and under different scenarios.
    - Testing scenarios that allow the assessment of UE battery consumption across the different offloading algorithms and parameterization and under different scenarios with and without duty cycling activated.
    - Testing scenarios that allow the assessment of the performance across the different offloading algorithms and parameterization when security features are activated in the prototype.

## 1.2 Scope of this Document

This deliverable is organized as follows:

- Section 2 describes the prototype that has been implemented and used during the testing, covering the MOTO platform and app, the Infrastructure API's, the offloading algorithms and the security framework.
- Section 3 covers the test plan, which has been split into functional and performance tests. Setups are also described in this section.
- In Section 4, testing results are summarised and analysed, collecting the different findings observed during the execution of the test plan.
- Finally, in section 5, main conclusions are listed.

# 2 Prototype description

The MOTO project proposes to design, implement, and evaluate an architecture taking full advantage of the latest advances in opportunistic networking to achieve efficient traffic offloading. Unlike many approaches in the literature, the MOTO architecture takes an operator point of view to opportunistic networking by keeping terminal-based offloading under the continuous control of the operator. The cellular infrastructure serves both as a control channel to track the content dissemination and as a last-resource option to deliver content.

In this prototype, we highlight the operation of the MOTO solution by using COTS hardware (server on X86 machine, eventually deployed remotely, and several smartphones running Android). The MOTO framework is also capable of interfacing with infrastructure operators to extract additional information useful to drive the offloading process (more details given in Section 2.4**Erreur ! Source du renvoi introuvable.**). We start giving a high-level description of the architecture (see also [1]), delving into the details of the implementation and demonstration scenario in the following sections.

The building blocks composing the MOTO architecture are highlighted in Figure 1 and detailed below. We consider in this prototype the case of a *push scenario*, where the content provider distributing the content knows in advance the list of users to whom the content should be forwarded. We implement the architecture *as a service* for operators and content providers. For this reason, we developed an architecture agnostic of the underlying cellular/Wi-Fi infrastructure tasked to serve data. Communications with operators and content providers are performed through a set of standard APIs that requires little integration effort.

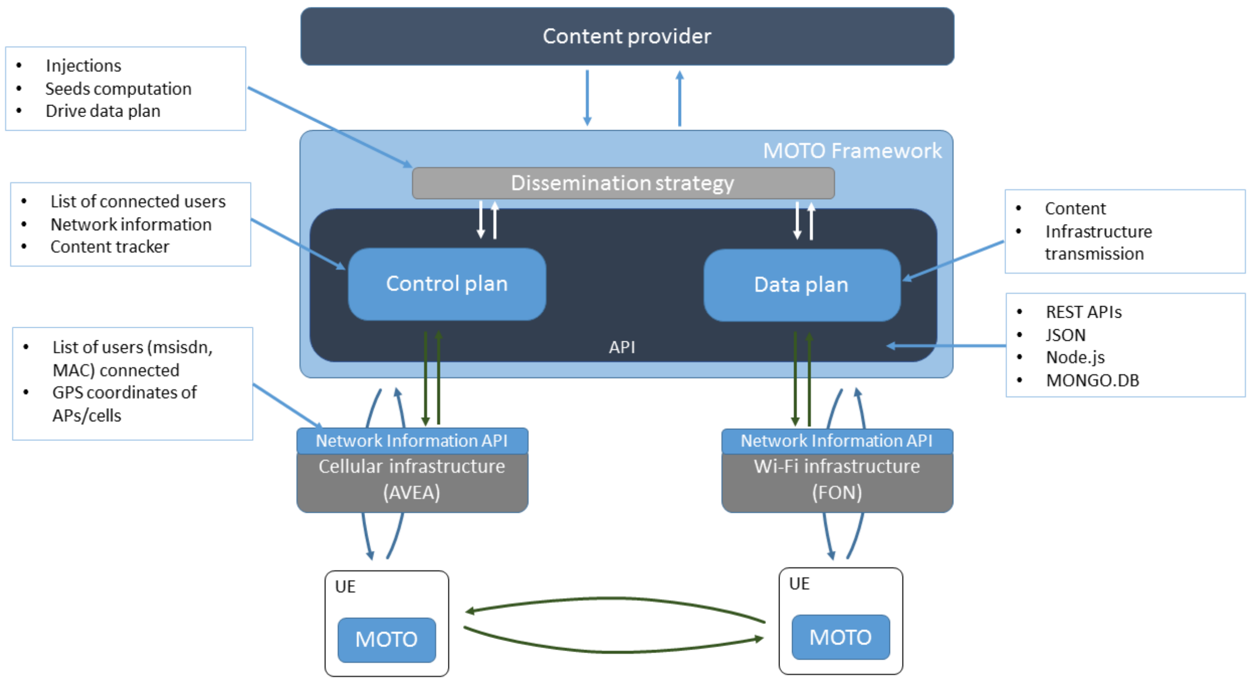

**Figure 1: MOTO high level prototype implementation.**

## 2.1 Content Provider

This block represents a content provider that has an agreement with the MOTO infrastructure. The content provider sends to the MOTO framework the content to be distributed, the list of users expecting the content, and their respective service-level agreement (SLA).

## 2.2 The MOTO Framework

Almost all the intelligence of the MOTO system is deployed within this block. In the prototype, the MOTO service is completely separated from the provider infrastructure (as depicted in Figure 1). This allows serving several operators by mutualizing MOTO computation resources. The choice calls for the implementation of standardized APIs in order to retrieve information from operator network.

**Control plan.** This module tracks the *content dissemination*, stores information on the *network status* (retrieved through network APIs) and the *user list*. It monitors the dissemination process by requiring acknowledgments from UEs when these latter receive the content. In this way, the dissemination strategy can be adjusted depending on the evolution of the dissemination. Information is passed to the dissemination strategy block in order to take the injection decisions. This module also performs data fusion on the identification of users, in order to assign each user a unique identifier to associate all the known information on different networks (Wi-Fi MAC address, MSISDN, and a unique MOTO Id).

**Dissemination strategy**. This functional block is responsible for piloting the offloading process. It determines, from the list of clients that require the content and from localization and topological information, the dissemination strategy to be applied. Injection algorithms are explained in detail in Section 2.5. The decisions regarding the injection and seeds selection are taken inside this block (e.g. deliver the content to clients A, B, and C through the cellular network and ask them to relay the content to their neighborhood using Wi-Fi).

**Data plan.** This block is in charge of piloting the actual transmission of the content on the infrastructure, by following the indications of the dissemination strategy block.

## 2.3 User Equipment

Two main types of services run at the user's terminal: MOTO services and built-in services (see **Erreur ! Source du renvoi introuvable.**).

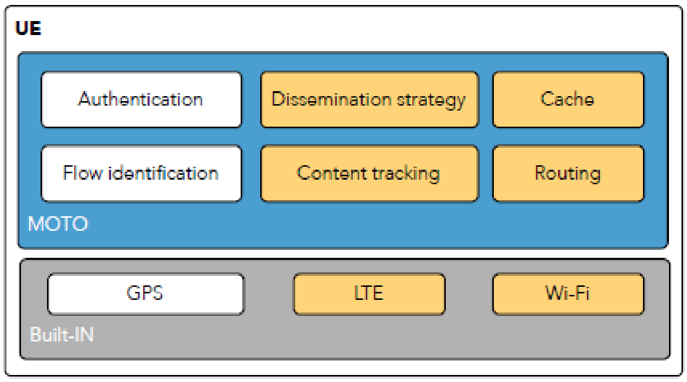

**Figure 2: MOTO high-level blocks at the user equipment.**

**MOTO services**. It is composed of several services that enable the user to benefit from the MOTO services. ***The authentication module*** is requested to authenticate MOTO customers with the MOTO platform and making sure that the security module is running correctly. ***The flow identification module*** is in charge of identifying each MOTO notification provided by the platform (new content, content feed ID, acknowledgments, etc.). ***The dissemination element*** broadcasts the ID of received content to its Wi-Fi neighborhood. If any customer is interested on this content, it will request it. The dissemination element opens a TCP connection and sends the content in unicast. The ***content tracking*** element is responsible for advertising the MOTO content tracker upon reception of MOTO content. ***The cache module*** requests the new content from the MOTO platform and stores it locally for future dissemination in D2D fashion.

Finally, the ***routing module*** (optional) is in charge of executing the routing policies given by the MOTO dissemination strategy block according to the role of the client and the content dissemination strategy.

**Built-in services**. It involves three elements inherently related to the operation of the node. The GPS element provides the geographic coordinates of the terminal. The LTE element allows the terminal to connect to an operator's LTE access network. The Wi-Fi element provides Wi-Fi connectivity to the terminal, either in infrastructure mode or in ad hoc mode (direct communications between terminals).

## 2.4 Infrastructure Operator

This block represents an infrastructure operator that has an agreement with the MOTO infrastructure. As described in [1], the MOTO framework retrieves information from the Infrastructure operator through the Infrastructure API. The infrastructure operator makes available for the MOTO framework the list of MOTO clients connected to each AP/enode-B and the network status information among others.

Despite having several methods defined for the Infrastructure API in [1], for the prototype only two of them are implemented: REQUEST_TOPO_INFO and REQUEST_NETWORK_STATUS. In the case of FON's Infrastructure API, the first one provides the MAC addresses of the users connected to the requested AP, while the second one provides network status information for a requested AP such as the number of users connected to the node, remaining capacity in the AP, etc.

In the case of AVEA's Infrastructure API, the 3G/LTE Core network status is polled periodically in order to get the list of each connected users at each requested cell. Through REQUEST_TOPO_INFO method, the list of connected users to the requested Cell-ID with unique identities such as MSISDN and IMSI is provided. This information can be obtained from operator's LTE core network element, MME. In order to provide MOTO framework with the required information that will enable the optimal offloading, the status and other related parameters of the 3G/LTE network status need to be obtained. The 3G/LTE network can provide MOTO framework with the number of connected users to each cell so that the offloading process is updated accordingly. Through REQUEST_NETWORK_STATUS method, network status information such as UP/DOWN status, number of connected users regarding the requested Cell-ID is provided. The response format to this request is in similar format to REQUEST_TOPO_INFO and includes information such as number of connected users to each Cell-ID.

## 2.5 Offloading algorithm

The algorithm must know the current number N of MOTO terminals interested in the content, the maximum delivery time D, and other algorithm-specific parameters. We assume that the algorithm also knows the instantaneous infection rate. This information is collected and made available by the MOTO platform in various forms as described in previous sections.

The following offloading algorithms are implemented in the prototype:

1. Initial injection: Only an initial number of users $I_0 \leq N$ are selected as initial seeds. When the elapsed time reaches the maximum delivery time D the algorithm triggers the panic zone re-injections.
2. Push-and-Track: The re-injection profile follows a fixed objective function $I^*(t) \in [0,1]$. The dissemination evolution is checked each $\Delta_t$. Each time the dissemination is below the selected objective function, the algorithm triggers a re-injection. Implemented objective functions are:
    i. Linear: $I^*(t) = t/D$.
    ii. Square-root: $I^*(t) = \sqrt{\frac{t}{D}}$.

   iii. Quadratic: $I^*(t) = \left[\frac{t}{D}\right]^2$.

3. DROID: The algorithm triggers re-injections when it discovers *plateaux* in content dissemination. Similarly to Push-and-track, the dissemination evolution is checked each $\Delta_t$. A moving window of size $W$ is employed. The algorithm re-injects additional copies whenever the rate of increase is below a certain threshold $\frac{I(t)-I(t-W)}{W} < \frac{1-I(t)}{D-t}$.

Initial injection is, in principle, the least complex algorithm and could be appropriate for simple scenarios when configured properly. On the other hand, DROID is the most complex one. It is able to adapt to rather simple scenarios and approximate the policy performance, while being also able to perform well in more complex cases, as shown in the simulation-based studies and in WP3 results [2].

Finally, the selection of which terminal to target can be random, where the content is pushed to a random node chosen uniformly among those that have not yet acknowledged reception, or it can depend on the geographical location of the users (deterministic). In the latter case, MOTO retrieves the connectivity information from the Infrastructure API. From this information, MOTO build a good enough picture of the global network topology to push content to a randomly chosen node within the largest uninfected group of users.

## 2.6 Security Framework

The security framework aims at ensuring the content integrity, privacy & confidentiality during the transmission process from end to end, meanwhile without holding back the performance and efficiency of the system. The following modules were implemented in order to achieve the aforementioned goals:

1. **Login module:** To enforce the authorization and authentication during the user login process. In addition to the conventional username and password method, the login information is also encrypted by the login timestamp with SHA1 algorithm. Thus, it prevents the attempted identity theft, together with a monitored login log from the security server.

2. **File signing module:** To ensure the content integrity and authenticated issuer of the file when MOTO is distributing the file through either platform to recipients or D2D communications. Thus, the security server is responsible for generating MOTO public key (Hex string) and private key (Hex string) by RSA-SHA1 algorithm which is a form of asymmetric cryptography to verify that the file is exactly what it claims to be. Next, the security server can sign their files with the MOTO private key within the limited usage time. And then the digital signature can be shared out by a REST call, as soon as the mobile user finished downloading the content. Afterwards, the mobile receivers can verify the digital signature by the corresponding MOTO public key obtained in the previous login process.

3. **File encryption & decryption module:** As this part is to prevent the content from erroneous content injection, it is optional if the transmission channel is secured such as HTTPS. Otherwise, the security server can encrypt the file with MOTO content encryption key (converted to 6 bytes AES key on the basis of MOTO public key) into unreadable byte array. Later, the android receivers verify the file signature, and then decrypt the file encrypted into the original file.

As a part of the MOTO prototype, the security approach can be integrated seamlessly within the MOTO platform in order to assure the success and optimal efficiency of the rest of MOTO services. Therefore, the security integration workflow is established as the following sequential diagram.

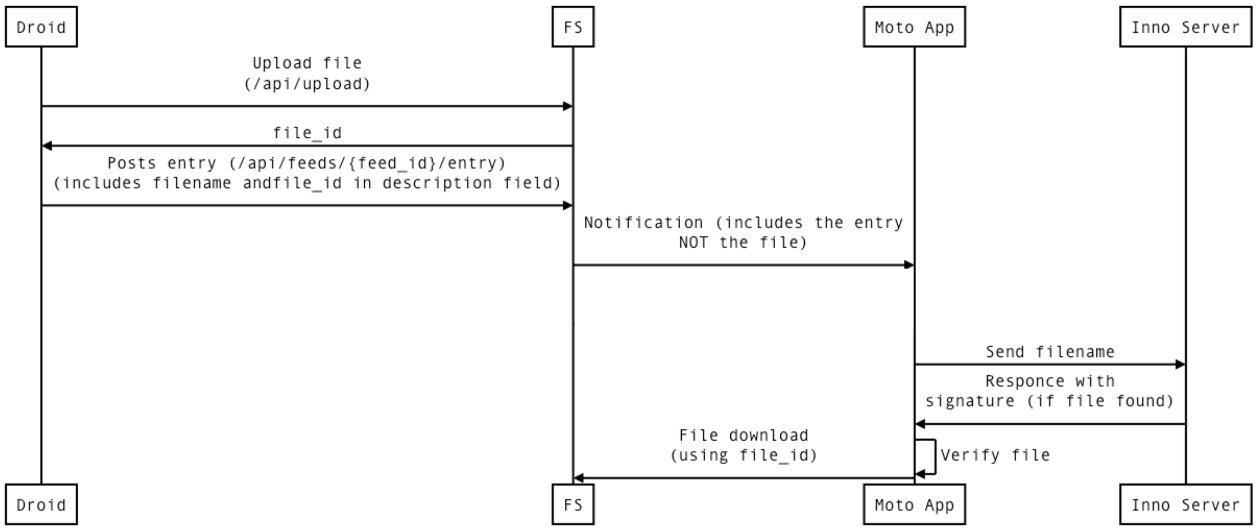
Droid
FS
Moto App
Inno Server
Upload file (/api/upload)
file_id
Posts entry (/api/feeds/{feed_id}/entry) (includes filename andfile_id in description field)
Notification (includes the entry NOT the file)
Send filename
Responce with signature (if file found)
File download (using file_id)
Verify file
Droid
FS
Moto App
Inno Server

**Figure 3. Security integration workflow.**

As shown in the workflow, the security server - INNO server is set separately from the web server, so as to protect the private and sensitive data with an extra layer of access control.  On the other hand, the security server only communicates with the MOTO mobile app once during the whole file transmission cycle. Moreover, the verification module is executed locally in the mobile device. Therefore, the interference with the main MOTO services is minimized. Due that the secured HTTPS channel has been set among the MOTO platform and the MOTO app, the file encryption and decryption module can be replaced by the bidirectional encrypted communication service offered by HTTPS.

# 3 Test plan

## 3.1 Testing scenarios

Two main testing scenarios were deployed in different locations. The most complete one has been deployed in Istanbul in AVEA premises (see Figure 4). It consists of an LTE and a 3G (AVEA) and a Wi-Fi (FON) networks collocated and providing overlapping coverage. A MOTO Platform is available at TCS premises. TCS would be acting as a MOTO Service Provider that offers MOTO services to FON and AVEA (operators). Thus, this setup also allows testing multi-operator scenarios. Operators involved might or might not have an agreement between them. It is not required since integration is done through a third party (i.e. TCS), the MOTO Service Provider. As described in section 4, this setup was used in the 2nd test fest hold in AVEA premises in Istanbul.

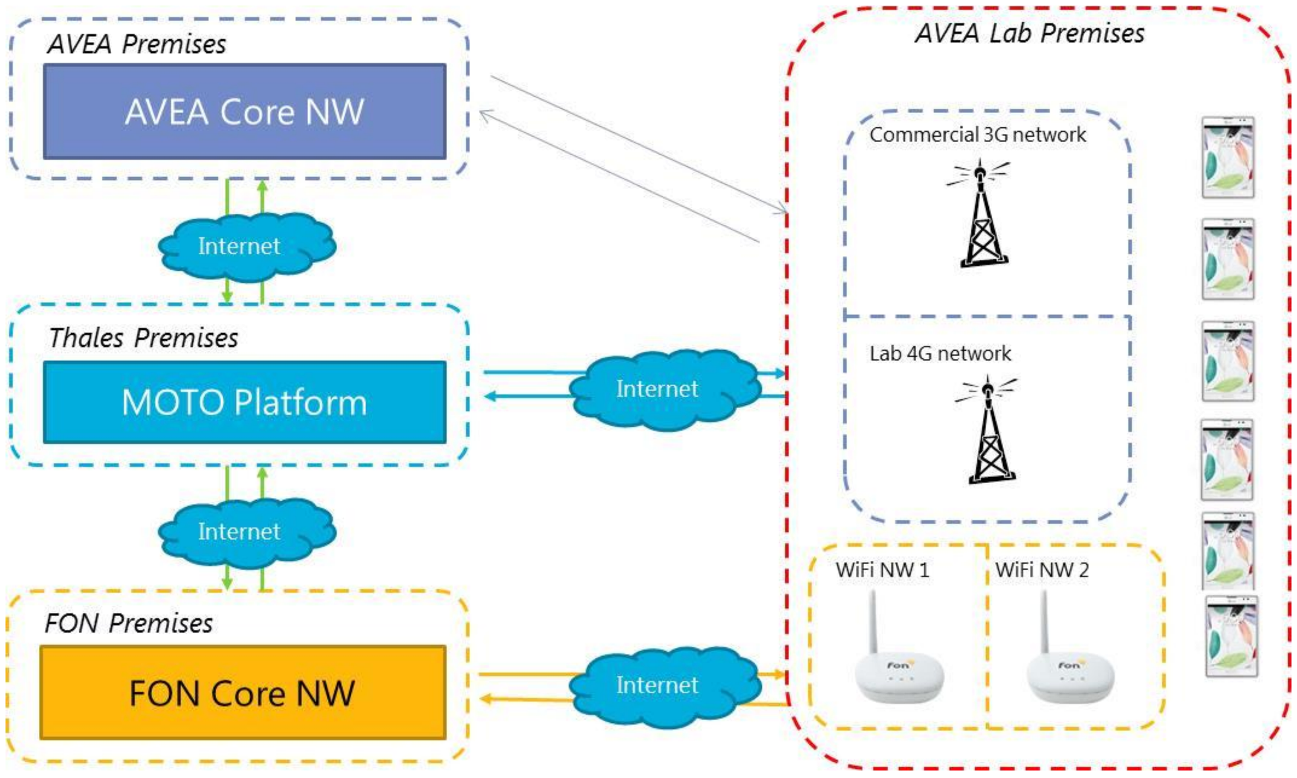

**Figure 4. Multi-operator scenario. 2nd Test Fest.**

This testing platform consists of the following main blocks:

- **3G (commercial) & 4G (lab) mobile networks:** The mobile core network was located in the same AVEA premises where the testing was performed. The 3G commercial network is usually congested during working hours due to AVEA's employees' generated traffic. This results into a scenario very close to reality, where 3G networks become congested due to an excessive number of simultaneous connections. During the analysis of the results, we will refer to 3G group to those devices attached to 3G network and to LTE group to those devices attached to the LTE one.

- **Wi-Fi network**: The Wi-Fi core network for the demo was located in FON premises (Spain), while the access points which gave service to the demo UEs where allocated in AVEA lab. The connection between the deployed access points and the Wi-Fi core network was performed through the Internet.

- **MOTO platform:** The MOTO platform was located in TCS premises (France). Both, the mobile and Wi-Fi core network were connected with the MOTO platform through the infrastructure API's over the Internet.

- **MOTO enabled handsets (UEs):** The UEs were connected to MOTO platform through the MOTO application over the Internet. These Handsets were provisioned with different connection capabilities, distributed among 3G, 4G, Wi-Fi NW1 and Wi-Fi NW2. As access points were used as relays for D2D communications (see section 3.2), having the handsets distributed between the two Wi-Fi networks was equivalent to having two groups of handsets split within two content dissemination areas (e.g. two different rooms in a museum). These handsets were also able to implement a Duty Cycle functionality, which could be activated / deactivated to assess energy efficiency improvement.

In addition to this one, an "only" Wi-Fi setup was deployed in FON Premises. It consists of a FON Wi-Fi network integrated with a MOTO Platform through the Infrastructure API plus a bundle of MOTO enabled devices. This MOTO Platform is also deployed in TCS Premises and accessible through the Internet. This setup also supports the use of commercial cellular networks (other than AVEA's, i.e. without any internal MOTO support and and without Infrastructure API), what can be useful for certain test cases. As mentioned in section 4, this setup was used in the 3rd test fest hold in FON premises in Getxo (Spain).

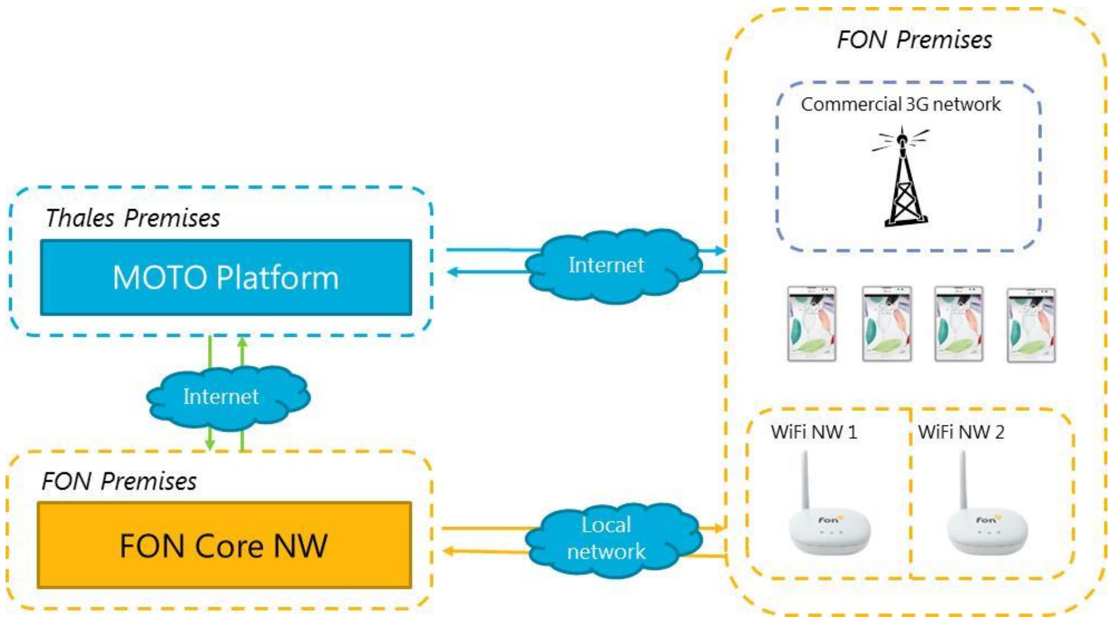

**Figure 5. "Only Wi-Fi" test setup. 3rd Test Fest.**

## 3.2 D2D emulation by using a Wi-Fi access point as a relay.

In the initial versions of the prototype, D2D communications were performed by using Wi-Fi in ad-hoc mode. While not supported officially, in older versions of Android (2.4 and below), it was possible to set the Wi-Fi interface in this mode. However, this functionality was removed due to security constraints (e.g., Google removed Wi-Fi ad-hoc from Android 3 onwards. iOS never supported it). Using old handsets with Android 2.4 versions in the testing was neither acceptable due to their obsolescence and lack of LTE support and resources.

From a requirement standpoint, appropriate lower layer technologies to transport D2D communications were not available at the time of the testing. For instance, both Bluetooth and Wi-Fi direct require pairing between handsets before starting to exchange content. This is not acceptable for opportunistic-based dissemination, mainly because of delay and signaling overheads introduced by the pairing process. This prevents also to scale-up the number of devices involved in dissemination because they consider a master-slave model preventing truly multi-hop communications. The discovery of neighboring devices is also severely limited by these shortcomings. Additionally, Wi-Fi direct is indeed emulating a Wi-Fi access point in the device delivering the data. It would be a similar case than using a Wi-Fi access point as a relay.

Due to the aforementioned limitations, we came back to use a standard Wi-Fi access point as a relay to handle D2D communications within the prototype, similarly to the “locally routed” scenario defined in 3GPP Rel. 13 ProSe functionality. Since MOTO is mostly focused on providing a service to operators, we considered this option as the best of the alternatives. Once D2D communications technologies are available e.g. by the 3GPP standardization, it will be straightforward to add the real D2D capability to current version of the prototype.

## 3.3 Test cases definition

Test cases are split into functional and performance tests.

### 3.3.1 Functional test cases

These test cases cover the validation of main functionalities, offloading algorithms and interfaces implemented in the prototype. This exercise allows us to ensure that the prototype is behaving as designed and, consequently, avoids introducing errors during the performance testing.

#### *3.3.1.1 Infrastructure API test cases*

This group of test cases is covering the validation of the methods regarding the Infrastructure API, implemented both in the MOTO platform and in the network sides.

**Table 2: Test IA1. Request network topology information.**

| | |
|---|---|
| **ID** | IA1 |
| **NAME** | Request network topology information |
| **DESCRIPTION** | This test verifies that the MOTO platform is able to retrieve network topology information of the user terminals connected to a Wi-Fi access point or an enodeB from the infrastructure operator through the Infrastructure API. |
| **SETUP** | This test requires setup defined in Figure 4. Multi-operator scenario. 2nd Test Fest.Figure 4. Alternatively, for Wi-Fi operator testing, setup depicted in Figure 5 is also valid.<br><br>MOTO Platform Infrastructure API client must have connectivity. |
| **PROCEDURE** | 1. The MOTO platform performs an HTTP GET through the Infrastructure API with the address /IDoperator/topology/IDdevice.<br>2. The operator will get the entries of the topology feed with “IDdevice” from its databases.<br>3. The operator returns to the MOTO platform the ATOM document that includes the list of UE connected to access point or the enodeB. |
| **EXPECTED RESULTS** | The MOTO platform consumes the ATOM document received from the operator through the Infrastructure API and acts accordingly. |

**Table 3: Test IA2. Request network status information.**

| | |
|---|---|
| **ID** | IA2 |
| **NAME** | Request network status information |
| **DESCRIPTION** | This test verifies that the MOTO platform is able to retrieve network status |

| | |
|---|---|
| | information from the infrastructure operator through the Infrastructure API. |
| **SETUP** | This test requires setup defined in Figure 4. Multi-operator scenario. 2nd Test Fest.Figure 4. Alternatively, for Wi-Fi operator testing, setup depicted in Figure 5 is also valid.<br><br>MOTO Platform Infrastructure API client must have connectivity. |
| **PROCEDURE** | 1. The MOTO platform performs an HTTP GET through the Infrastructure API with the address /IDoperator/netstats/IDdevice.<br>2. The operator will get the entries of the network status feed with "IDdevice" from its databases.<br>3. The operator returns to the MOTO platform the ATOM document that includes the WiFi access point or the enodeB status (e.g. No of UEs connected, a.remaining capacity, etc.). |
| **EXPECTED RESULTS** | The MOTO platform consumes the ATOM document received from the operator through the Infrastructure API and acts accordingly. |

#### *3.3.1.2 Control Channel cases*

This group of test cases is covering the validation of the methods regarding the control channel (i.e. communications between the MOTO platform and the terminals), implemented both in the MOTO platform and in the terminal sides.

**Table 4: Test CC1. Control message reception.**

| | |
|---|---|
| **ID** | CC1 |
| **NAME** | Control message reception |
| **DESCRIPTION** | This test verifies that the MOTO platform is capable of receiving acknowledgements on data reception and contextual information from MOTO terminals. |
| **SETUP** | This test requires setup defined in Figure 4. Multi-operator scenario. 2nd Test Fest.Figure 4. Alternatively, setup depicted in Figure 5 is also valid.<br><br>MOTO terminals must have cellular connectivity and they must be authenticated by the MOTO platform as proper MOTO users. |
| **PROCEDURE** | 1. Multiple MOTO terminals send a CONTENT_REQUEST message for the content with *msg_id* to the MOTO Platform.<br>2. The MOTO Platform sends the content to one of the requesters MOTO terminal.<br>3. The MOTO terminal sends a CONTENT_RECEIVED message on the **cellular** interface to */feedsync/rest/myfeeds/4/entries/msg_id/Ack.*<br>4. The MOTO terminal transmits data to neighbor MOTO terminals using **Wi-Fi ad hoc** connectivity.<br>5. Upon data reception, the MOTO terminals send a CONTENT_RECEIVED message on the **cellular** interface to */feedsync/rest/myfeeds/4/entries/msg_id/Ack.* |

| | |
|---|---|
| **EXPECTED RESULTS** | The MOTO platform consumes the CONTENT_RECEIVED messages received from terminals. The MOTO platform identifies the MOTO users using the information received through the Infrastructure API's. The MOTO platform updates its ACK feed accordingly. |

#### 3.3.1.3 *Offloading algorithms test cases*

This group of test cases is covering the functional validation of the offloading algorithms implemented in the prototype. It also includes the testing of involved terminal to terminal communications.

**Table 5: OA1. Panic zone trigger.**

| | |
|---|---|
| **ID** | OA1 |
| **NAME** | Panic zone trigger |
| **DESCRIPTION** | This test verifies that the MOTO platform assures a maximum delivery delay using panic zone re-injections. |
| **SETUP** | This test requires setup defined in Figure 4. Multi-operator scenario. 2nd Test Fest.Figure 4. Alternatively, setup depicted in Figure 5 is also valid.<br>MOTO terminals do not have any other alternative communication interface than the cellular interface (Wi-Fi ad hoc is shut off).<br>We must define the parameter D (Delay tolerance before the panic zone). |
| **PROCEDURE** | 1. Multiple MOTO terminals send a CONTENT_REQUEST message with *msg_id* to the MOTO Platform.<br>2. The MOTO Platform sends the content to one of the requesters MOTO terminal.<br>3. When the time elapsed after the first transmission reaches **D**, the MOTO platform re-injects the content to all the requesters on the **cellular** interface. |
| **EXPECTED RESULTS** | All the MOTO terminals except one receive the content on the cellular interface soon after the time D has expired. |

#### 3.3.1.4 *Security test cases (Ed. INNO)*

This group of test cases is covering the validation of those security functions implemented in the prototype.

**Table 6: SEC1. Authentication (login).**

| | |
|---|---|
| **ID** | SEC1 |
| **NAME** | Authentication in the MOTO mobile app |
| **DESCRIPTION** | This test verifies that the MOTO user is able to authenticate in the MOTO mobile app against the MOTO platform to start using MOTO services |
| **SETUP** | This test requires setup defined in Figure 4. Multi-operator scenario. 2nd Test Fest.Figure 4. Alternatively, setup depicted in Figure 5 is also valid.<br>MOTO platform must be reachable from the UE (coverage range). |

| | |
|---|---|
| **PROCEDURE** | **1.** The MOTO user performs a HTTP / HTTPS request to the MOTO platform to access MOTO services. <br> **2.** The MOTO platform will request user credentials <br> **3.** UE will authenticate in front of MOTO. <br> **4.** MOTO platform will enable MOTO user to participate |
| **EXPECTED RESULTS** | The UE who successfully authenticates in front of the MOTO platform is able to participate in the MOTO services |

**Table 7: SEC2. Signature implementation and verification.**

| | |
|---|---|
| **ID** | SEC2 |
| **NAME** | Signature implementation and verification |
| **DESCRIPTION** | This test verifies that the MOTO platform is able to sign the content and that the MOTO application is able to verify the MOTO signature |
| **SETUP** | This test requires setup defined in Figure 4. Multi-operator scenario. 2nd Test Fest.Figure 4. Alternatively, setup depicted in Figure 5 is also valid. <br> MOTO platform must be reachable from the UE (coverage range) and user must authenticate against MOTO platform. |
| **PROCEDURE** | **1.** The MOTO user performs a HTTP / HTTPS request to the MOTO platform to request some content <br> **2.** The MOTO platform will sign the content with the MOTO signature <br> **3.** UE receives the content and is able to verify MOTO signature |
| **EXPECTED RESULTS** | MOTO is able to sign a content with its own signature and the MOTO UE is able to verify the signature of the received content |

**Table 8: SEC3. Content encryption.**

| | |
|---|---|
| **ID** | SEC3 |
| **NAME** | Content encryption |
| **DESCRIPTION** | This test verifies that the MOTO platform is able to encrypt the MOTO content before sending it to the users and that the users are able to decrypt this content |
| **SETUP** | This test requires setup defined in Figure 4. Multi-operator scenario. 2nd Test Fest.Figure 4. Alternatively, setup depicted in Figure 5 is also valid. <br> MOTO platform must be reachable from the UE (coverage range) and user must authenticate against MOTO platform. |
| **PROCEDURE** | **1.** The MOTO user performs a HTTP / HTTPS request to the MOTO platform to request some content <br> **2.** The MOTO platform will encrypt he content before sending it to the users <br> 3. UE receives the encrypted content and is able to decrypt it to see the |

| | |
|---|---|
| | real content |
| **EXPECTED RESULTS** | MOTO content is encrypted before being sent and users decrypt this content |

### 3.3.2 <u>Performance test cases</u>

This section is covering the test cases for the assessment of the performance of the MOTO framework. Main objective is to emulate testing scenarios as realistic as possible in order to derive the performance under actual scenarios.

#### *3.3.2.1 Metrics*

This testing resulted in the collection of the following metrics across the different tests and under different offloading algorithms parameterization and conditions:

- **Delay from first injection:** for each client involved in the dissemination, the elapsed time between first injection and final reception by the client. This metric is significant for the assessment of the QoS and will be evaluated according to delay tolerance parameters.
- **Panic zone ratio:** the ratio between number of injections that reach the panic zone (i.e. surpass maximum delay threshold) and the total number of injections. This metric will help to assess how likely is to reach the panic zone under different scenarios.
- **Offloading gain:** the ratio between successfully completed D2D injections and total number of injections, including those directly executed through the WAN. This metric will allow assessing the amount of traffic that is effectively offloaded through T2T communications. In the optimal (ideal) case, the offloading gain would be N-1/N where N is the number of clients, meaning only 1 injection would be sufficient to reach all intended destinations.
- **End-user terminal battery consumption:** collection of terminal battery consumption to assess battery savings enabled by MOTO framework.

#### *3.3.2.2 Initial Injection*

**Table 9: PER1. Initial injection algorithm assessment.**

| | |
|---|---|
| **ID** | PER1 |
| **NAME** | Initial Injection offloading algorithm assessment. |
| **DESCRIPTION** | This test case covers the assessment of Initial injection offloading algorithm under a static pattern: all handsets in the same location (e.g. stadium use case) distributed in different dissemination areas (e.g. museum use case).<br><br>Duty cycling will be evaluated.<br><br>Security features evaluated.<br><br>Additionally, different parameterization will be applied to perform a sensitivity analysis. As far as possible, all metrics defined in section 3.3.2.1 should be collected. |
| **SETUP** | This test requires setup defined in Figure 4. Multi-operator scenario. 2nd Test Fest.Figure 4. Alternatively, setup depicted in Figure 5 is also valid, when mobile network connectivity is not required.<br><br>• Offloading algorithms used: Initial injection |

| | |
|---|---|
| | • Different $I_0$ as a percentage of the number of users expecting to receive the content<br>• Duty cycling: ON & OFF.<br>• Security features: ON & OFF. |
| **PROCEDURE** | 1. MOTO platform is configured according to the setup description.<br>2. Injection occurs to the $I_0$ selected nodes wait until all deliveries have been completed.<br>3. Repeat steps #1 - #3 6 times per group of parameter values.<br>4. Repeat steps #1 - #4 under the two mobility patterns.<br>5. Repeat steps #1 - #4 with duty ciclying ON. |
| **EXPECTED RESULTS** | The test allows us collecting a meaningful bundle of metrics for performance validation of Initial injection offloading algorithm under diferent mobility scenarios and parameterization. |

#### 3.3.2.3 *Push & Track*

**Table 10: PER2. Push & Track algorithm assessment.**

| | |
|---|---|
| **ID** | PER2 |
| **NAME** | Push & Track algorithm assessment. |
| **DESCRIPTION** | This test case covers the assessment of Push & Track offloading algorithm under a static pattern: all handsets in the same location (e.g. stadium use case) distributed in different dissemination areas (e.g. museum use case).<br>Duty cycling will be evaluated.<br>Security features evaluated.<br>Additionally, different parameterization will be applied to perform a sensitivity analysis. As far as possible, all metrics defined in section 3.3.2.1 should be collected. |
| **SETUP** | This test requires setup defined in Figure 4. Multi-operator scenario. 2nd Test Fest.Figure 4. Alternatively, setup depicted in Figure 5 is also valid, when mobile network connectivity is not required.<br>• Offloading algorithms used: Push & Track<br>• Reinjection profile function: linear, square root, and quadratic.<br>• Mobility patterns: emulated and static.<br>• Duty cycling: ON & OFF.<br>• Security features: ON & OFF. |
| **PROCEDURE** | 1. MOTO platform is configured according to the setup description with duty cycling OFF.<br>2. Push&Track algorithm is started to control the initial injection and the subsequent re-injections. |

| | |
|---|---|
| | 3. Wait until all deliveries have been completed.<br>4. Repeat steps #1 - #3 6 times per group of parameter values.<br>5. Repeat steps #1 - #4 under the two mobility patterns.<br>6. Repeat steps #1 - #4 with duty ciclying ON. |
| **EXPECTED RESULTS** | The test allows us collecting a meaningful bundle of metrics for performance validation of Push & Track offloading algorithm under diferent mobility scenarios, duty cycling and parameterization. |

### *3.3.2.4 DROID*

**Table 11: PER3. DROID algorithm assessment.**

| | |
|---|---|
| **ID** | PER3 |
| **NAME** | DROID offloading algorithm assessment. |
| **DESCRIPTION** | This test case covers the assessment of DROID offloading algorithm under a static pattern: all handsets in the same location (e.g. stadium use case) distributed in different dissemination areas (e.g. museum use case).<br>Duty cycling will be evaluated.<br>Security features evaluated.<br>Additionally, different parameterization will be applied to perform a sensitivity analysis. As far as possible, all metrics defined in section 3.3.2.1 should be collected. |
| **SETUP** | This test requires setup defined in section **Erreur ! Source du renvoi introuvable.**. The configuration and parameterization of the MOTO framework is the following:<br>• Offloading algorithms used: DROID.<br>• Different moving windows W.<br>• Mobility patterns: emulated and static.<br>• Duty cycling: ON & OFF.<br>• Security features: ON & OFF. |
| **PROCEDURE** | 1. MOTO platform is configured according to the setup description.<br>2. Droid algorithm is started to control the initial injection and the subsequent re-injections Wait until all deliveries have been completed.<br>3. Repeat steps #1 - #3 6 times per group of parameter values.<br>4. Repeat steps #1 - #4 under the two mobility patterns.<br>5. Repeat steps #1 - #4 with duty ciclying ON. |
| **EXPECTED RESULTS** | The test allows us collecting a meaningful bundle of metrics for performance validation of DROID offloading algorithm under diferent mobility scenarios and parameterization. |

# 4 Results & Analysis

The execution of the tests described in section 3 was performed across three different test fests. 1st and 2nd ones were done in Istanbul in AVEA premises, while the third one was arranged in Getxo (Spain) in FON premises. 1st test fest (April 2015) was mainly devoted for MOTO different modules integration activities, covering mainly functional testing. In the 2nd test fest (July 2015), a more advanced prototype was available. Setup described in Figure 4 was used, trying to emulate a multi-operator scenario as real as possible. During this test fest a complete bundle of performance testing was executed, obtaining the first performance results across offloading algorithms and different parameterizations. However, some tests on security features could not be completed. Those were performed during 3rd test fest (September 2015), having, at this point of time, all the necessary components to perform an appropriate analysis.

In the following section, results gathered across these three test fests are described and analyzed.

## 4.1 Functional test cases results & analysis

Functional testing was performed during the three test fests, as new features and integrations were being done. This testing is also necessary to ensure that the prototype is working well prior to start with the performance testing. In the following tables, results of the functional testing across the test fests are summarized. At the last test fest, the prototype had security features and Infrastructure API's fully integrated and passed all the functional tests. Below a summary of the functional tests executed during the three test fest:

**Table 12. 1st Test Fest - Functional testing results.**

| Test Title | Pass / Fail | Comments |
|---|---|---|
| IA1 - Request network topology information (FON) | Pass | The information is properly obtained from the operator's API and sent to the MOTO Platform. No integration with Feedsync. |
| IA2 - Request network topology information (FON) | Pass | The information is properly obtained from the operator's API and sent to the MOTO Platform. No integration with Feedsync. |
| CC1 - Control message reception | Pass | The MOTO terminals send a CONTENT_RECEIVED message on the cellular interface to /feedsync/rest/myfeeds/4/entries/msg_id/Ack correctly. |
| OA1 - Panic zone trigger | Pass | All the MOTO terminals except one receive the content on the cellular interface soon after the time D has expired. |
| IA1 - Request network topology information (AVEA) | Not performed | |
| IA2 - Request network topology information (AVEA) | Not performed | |
| SEC1 Certificate delivery | Not performed | |
| SEC2 Feedback acknowledgement | Not performed | |

**Table 13. 2nd Test Fest - Functional testing results.**

| Test Title | Pass / Fail | Comments |
|---|---|---|
| IA1 - Request network topology information (FON) | Pass | Integrated with Feedsync |
| IA2 - Request network topology information (FON) | Pass | Integrated with Feedsync |
| CC1 - Control message reception | Pass | Some issues with the MOTO app have been solved during the test fest |
| OA1 - Panic zone trigger | Pass | |
| IA1 - Request network topology information (AVEA) | Pass | Integrated with Feedsync |
| IA2 - Request network topology information (AVEA) | Pass | Integrated with Feedsync |
| SEC1 Authentication in the MOTO mobile app | Pass | Isolated testing. Not integrated with MOTO app / Feedsync |
| SEC2 Signature implementation and verification | Pass | Isolated testing. Not integrated with MOTO app / Feedsync |
| SEC3 Content encryption | Pass | Isolated testing. Not integrated with MOTO app / Feedsync |

**Table 14. 3rd Test Fest - Functional testing results.**

| Test Title | Pass / Fail | Comments |
|---|---|---|
| IA1 - Request network topology information (FON) | Pass | Integrated with Feedsync |
| IA2 - Request network topology information (FON) | Pass | Integrated with Feedsync |
| CC1 - Control message reception | Pass | Some issues with the MOTO app have been solved during the test fest |
| OA1 - Panic zone trigger | Pass | |
| IA1 - Request network topology information (AVEA) | Pass | Integrated with Feedsync |
| IA2 - Request network topology information (AVEA) | Pass | Integrated with Feedsync |
| SEC1 Authentication in the MOTO mobile app | Pass | |
| SEC2 Signature implementation and verification | Pass | |
| SEC3 Content encryption | Pass | After including HTTPS, encryption through an external security component is not required. |

## 4.2 Performance test cases results & analysis

This section is covering the test cases conducted in order to assess the performance of the MOTO framework. The main objective was to emulate testing scenarios as realistic as possible in order to derive the performance under actual scenarios. These tests were performed during 2nd and 3rd test fests.

### 4.2.1 Algorithms performance assessment

This testing was performed under 2nd test fest multi-operator scenario depicted in Figure 4. 10 MOTO enabled handsets were used under static pattern conditions and within two separated Wi-Fi networks (i.e. emulating two dissemination areas, e.g. two rooms of a museum). Specifically, the 10 clients were split in two groups: 6 handsets in one group and 4 in the other group. All the 10 clients were subscribed to the same content, to be distributed via the MOTO platform. The delivery deadline was set to 30 or 60 seconds, with a panic zone of 5 seconds.

For the sake of having meaningful statistical results, for each of the algorithms and configurations, several iterations were executed. This way, the results obtained were checked against several independently conducted tests (iterations).

The main metric to maximize is the offloading gain, since offloading is the main purpose of the designed algorithms. However, in order to perform a realistic testing, and comply with the expected MOTO approach behavior (QoE), the offloading must comply with certain conditions; Firstly, the stated maximum delay must not be surpassed, in order to comply with the terms of the content provider using MOTO. Secondly, the use of MOTO should avoid the unnecessary consumption of end user device resources. In the tests, we considered battery life as the most relevant user's device resource.

Also note that the order, which the algorithms are analyzed in the following section, is based on their complexity, starting with the least complex (Initial Injection) to the most complex one (DROID). Initial Injection

The main aim of this performance test was to assess Initial Injection algorithm. To do so, the following parameters were set:

**Table 15. Initial injection performance testing - scenario details.**

| **Initial Injection** | |
|---|---|
| **Test Case ID:** PER1 | **Delay Tolerance:** 30s |
| **Test Priority (Low/Medium/High):** HIGH | **Panic Zone Duration:** 5s / **Content size:** 200kb |
| **Test Title:** Initial Injection algorithm assessment. | **Number of MOTO enabled users:** 10 |
| **Description:** This test case covers the assessment of Initial Injection offloading algorithm under a WAN (LTE / 3G or Wi-Fi). | **Reinjection Profile:** N/A |
| | **$I_0$ as a percentage of N. $I_0$:** 10%, 20%, 40% |
| | **Moving window W:** N/A |
| | **Mobility Pattern:** Static |

**Other comments:**

There are two dissemination areas separated by Wi-Fi networks. In some tests, injection strategy considered this fact (information retrieved from the Infrastructure API) – deterministic injection strategy. In others, the injection was random (i.e. no information regarding how the users were split within the two groups was considered by the injection strategy).

Several repetitions have been performed for each the combinations.

**Table 16. Initial injection. Results summary.**

| Configuration | Average Delay from first injection | StdDev Delay from first injection | Panic Zone ratio | Average Offloading gain |
|---|---|---|---|---|
| Deterministic - 10% | 12.59s | 12.12s | 40% | 50% |
| Random - 10% | 14.86s | 11.94s | 48% | 42% |
| Deterministic - 20% | 5.78s | 7.31s | 0% | 80% |
| Random - 20% | 11.10s | 11.05s | 32% | 48% |
| Deterministic - 40% | 4.05s | 3.47s | 0% | 60% |
| Random - 40% | 3.84s | 2.67s | 0% | 60% |

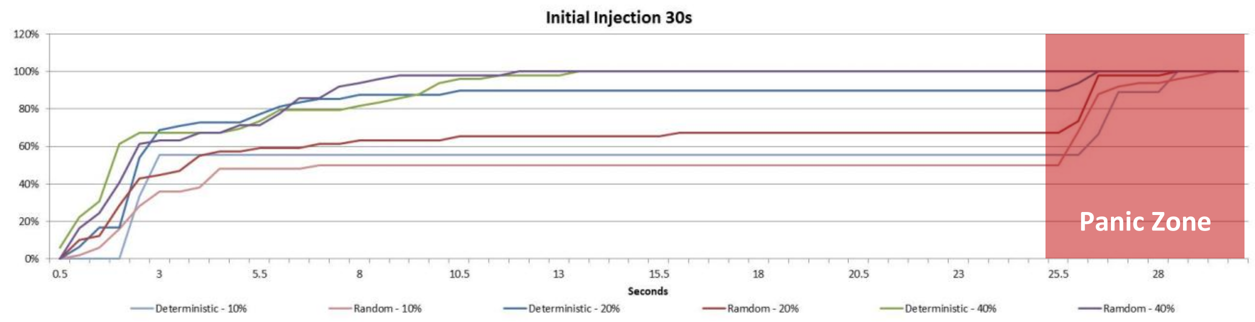

**Figure 6. Dissemination evolution - Initial Injection - 30 secs delay tolerance - 5 secs panic zone.**

The following observations are collected:

- Initial Injection is the simplest offloading algorithm. Differently from Push & track and DROiD, only initial copies are injected through the cellular network. Additional copies are not injected until panic zone is reached. This is the expected behavior of Initial Injection algorithm.
- We notice a threshold effect on the number of initial injections: injecting only one copy at the beginning (10%) is not enough since nodes are static within their respective dissemination areas. Consequently, one of the groups will always reach the panic zone. 20% of initial seeds configuration shows the best performance results, especially for the deterministic strategy (one copy inside the LTE group1 and one copy inside the 3G group2). Additional injections (e.g. 40%) are unnecessary, lowering the overall offloading efficiency (capabilities of D2D communications are underestimated).

[1] LTE group: devices attached to AVEA's LTE lab network.

[2] 3G group: device attached to AVEA's 3G commercial network.

- Under Initial Injection, a deterministic choice of seeds targeting both the dissemination areas provides better results than random. In the proposed scenario, considering two initial copies and random injections, we have in average the 46% of probability of targeting the same dissemination area with both copies, lowering in those cases the average offloading efficiency. This is reflected in experimental results with a drop in the offloading efficiency from 80% to 48% under these two strategies. This stresses the importance of the information retrieved from the operator network through the Infrastructure API's in order to drive the offloading process (in particular the choice of seed nodes).

#### *4.2.1.1 Push & Track*

The main aim of this performance test was to assess the performance of Push & Track algorithm. To do so, the following parameters were set:

**Table 17. Push & Track performance testing - scenario details.**

| **PUSH & TRACK** | |
|---|---|
| **Test Case ID:** PER2 | **Delay Tolerance:** 60s, 30s |
| **Test Priority (Low/Medium/High):** HIGH | **Panic Zone Duration:** 5s / **Content size:** 200kb |
| **Test Title:** Push & Track algorithm assessment. | **Number of MOTO enabled users:** 10 |
| **Description:** This test case covers the assessment of Push & Track offloading algorithm under a WAN (LTE / 3G or Wi-Fi). | **Reinjection Profile:** Linear, Square Root & Quadratic |
| | **$I_0$ as a percentage of N. $I_0$:** N/A |
| | **Moving window W:** N/A |
| | **Mobility Pattern:** Static |
| **Other comments:** There are two dissemination areas separated by Wi-Fi networks. In some tests, injection strategy considered this fact (information retrieved from the Infrastructure API) – deterministic injection strategy. In others, the injection was random (i.e. no information regarding how the users were split within the two groups was considered by the injection strategy). Several repetitions have been performed for each the combinations. | |

**Table 18.- Push & Track. Results summary.**

| Configuration | Average Delay from first injection | StdDev Delay from first injection | Panic Zone ratio | Average Offloading gain |
|---|---|---|---|---|
| Deterministic - Linear - 60s | 17.30s | 18.47s | 2% | 72% |
| Deterministic - Linear - 30s | 11.92s | 11.03s | 6% | 64% |
| Random - Square – 30s | 4.19s | 2.50s | 0% | 62% |
| Random - Linear - 30s | 8.56s | 5.52s | 4% | 66% |
| Random – Quadratic - 30s | 13.05s | 11.00s | 12% | 61% |

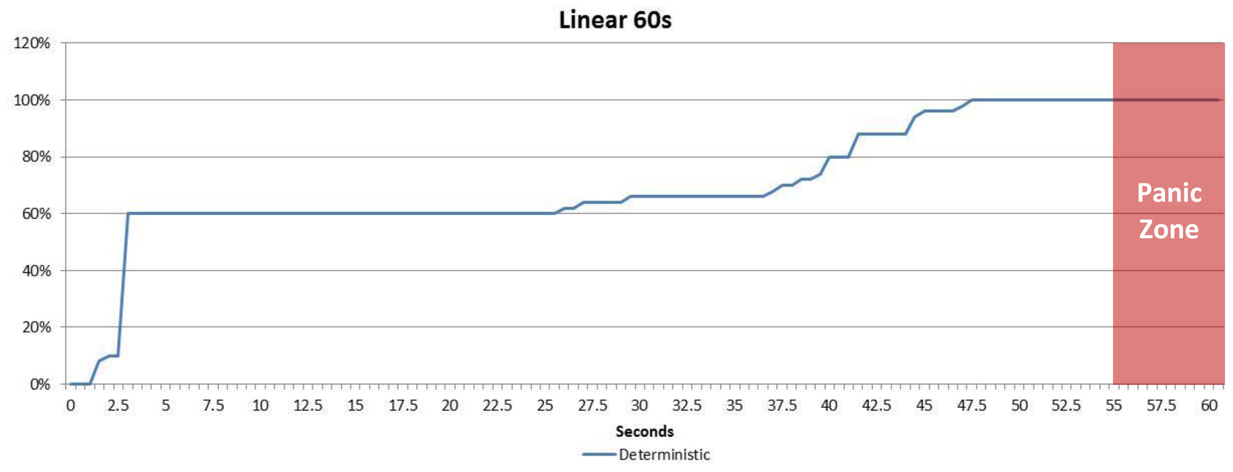

**Figure 7. Dissemination evolution - Push & track - Linear - 60 secs delay tolerance - 5 secs panic zone.**

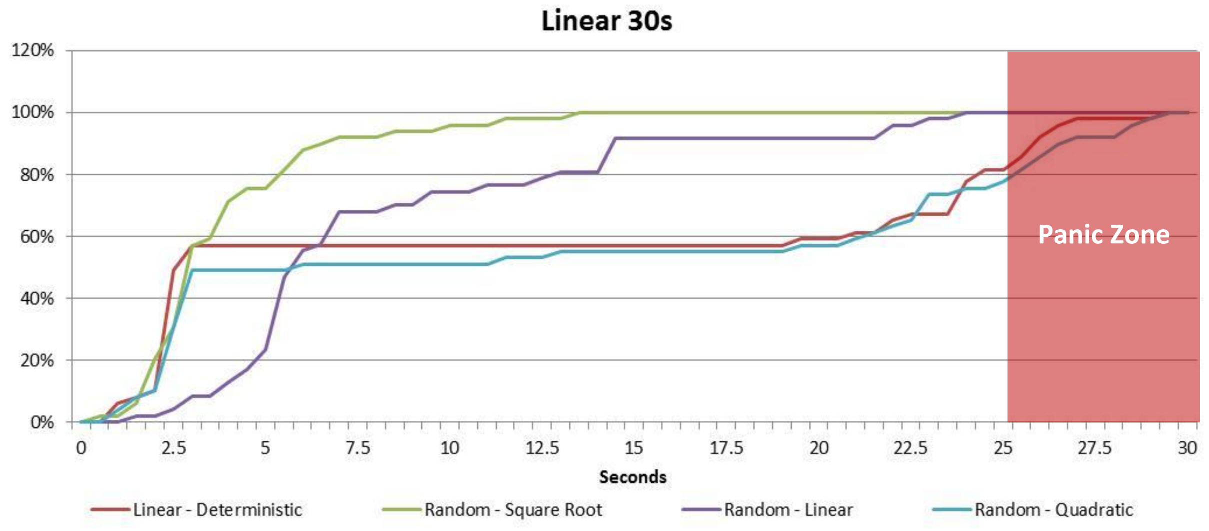

**Figure 8. Dissemination evolution - Push & track - 30 secs delay tolerance - 5 secs panic zone.**

After analyzing the results, the following observations have been collected:

- Offloading algorithms work as designed. They follow a more or less aggressive injection strategy depending on the objective function (i.e. following the discussion in section 2.5, *quadratic* is less aggressive than *square root*) and the evolution of content dissemination. Remaining content is injected through the WAN when reaching the panic zone, to guarantee the maximum reception delay.
- Longer delay tolerance provides better offloading results. Longer delay tolerance allows exchanging more content on the opportunistic channel. As expected, services with more relaxed delay requirements will allow higher offloading efficiency.
- Deterministic (i.e. based on infrastructure API information) injection does not show significant improvements in terms of aggregate performance over random. While we could expect better results for the deterministic case where Infrastructure API information is considered, this does not happen. The reason is that Push & track strategies consider only the overall evolution of the content. In many cases, it estimates enough to be above the threshold. This behavior can be evaluated in Figure 8**Erreur ! Source du renvoi introuvable.** A single injection at the beginning is enough for a fast dissemination inside the first group (60% of nodes). Then, no re-injections are triggered until 20s, because content dissemination

stays above the objective function. However, we can notice the presence of a long plateau due to the impossibility of contacts between the two groups (the already infected and the uninfected). This shortcoming confirms the outcomes of simulations and calls for more adaptive offloading strategies (i.e., DROID).

- Among the reinjection profiles, quadratic shows slightly worse results. In effect, this is the least aggressive injection strategy. In case of very short delay tolerances (such as 30s), re-injections with this strategy happen too late, not giving enough time to devices to exchange data in D2D fashion. It results an increased amount of panic re-injections. Again, a more adaptive strategy (i.e., without a predefined injection strategy) could help coping with this problem.

- From the point-of-view of MOTO operator, D2D dissemination inside the *LTE group*[3] is much faster (almost 0.5s for each node) than the *3G group*[4] (4 to 5s for each node). There is not a conclusive reason for this behavior. Our understanding is that terminals with LTE capabilities are more recent and equipped with more advanced wireless cards. Another explanation might be related to the fact that, to reach the MOTO platform, the ACKs are always routed through the cellular network. The 3G network tested in the 2nd test fest was the AVEA commercial network, which was highly congested during working hours. Conversely, the 4G network was a test network without commercial users. While ACKs are in general very small, the different congestion levels could influence the access time of the two networks.

### *4.2.1.2 DROID*

The main aim of this performance test was to assess DROID algorithm. To do so, the following parameters were set:

**Table 19. DROID performance testing - scenario details.**

| **DROID** | |
|---|---|
| **Test Case ID:** PER3 | **Delay Tolerance:** 30s |
| **Test Priority (Low/Medium/High):** HIGH | **Panic Zone Duration:** 5s / **Content size:** 200kb |
| **Test Title:** DROID algorithm assessment. | **Number of MOTO enabled users:** 10 |
| **Description:** This test case covers the assessment of DROID offloading algorithm under a WAN (LTE/3G or Wi-Fi). | **Reinjection Profile:** N/A |
| | **$I_0$ as a percentage of N. $I_0$:** N/A |
| | **Moving window W:** 5s, 3s, 2s |
| | **Mobility Pattern:** Static |

[3] LTE group: devices attached to AVEA's LTE lab network.

[4] 3G group: device attached to AVEA's 3G commercial network.

**Other comments:**

There are two dissemination areas separated by Wi-Fi networks. In some tests, injection strategy considered this fact (information retrieved from the Infrastructure API) – deterministic injection strategy. In others, the injection was random (i.e. no information regarding how the users were split within the two groups was considered by the injection strategy).

Several repetitions have been performed for each the combinations.

**Table 20. DROID. Results summary.**

| Configuration | Average Delay from first injection | StdDev Delay from first injection | Panic Zone ratio | Average Offloading gain |
|---|---|---|---|---|
| Deterministic - W 5s | 7.91 | 7.76 | 2% | 72% |
| Random - W 5s | 11.01 | 12.58 | 6% | 64% |
| Deterministic - W 3s | 6.24 | 5.43 | 0% | 72% |
| Random - W 3s | 8.23 | 6.07 | 0% | 73% |
| Deterministic - W 2s | 6.65 | 5.44 | 0% | 64% |
| Random - W 2s | 6.19 | 5.77 | 0% | 68% |

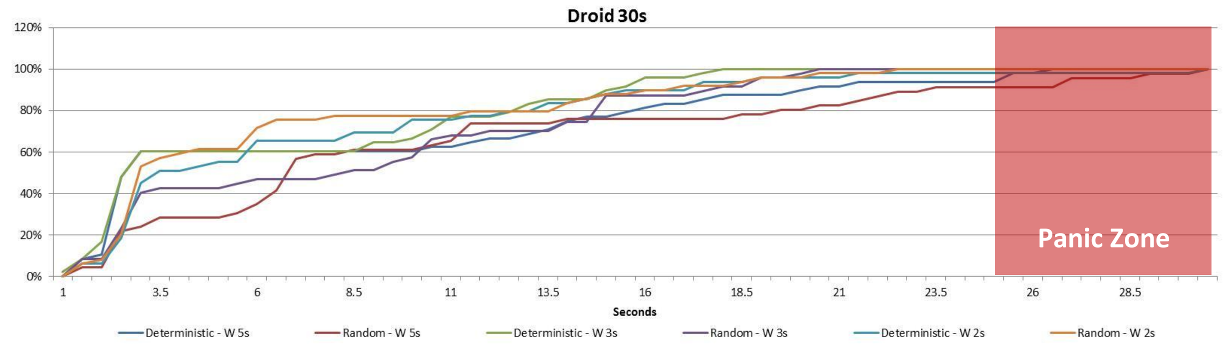

**Figure 9. Dissemination evolution - DROID - 30 secs delay tolerance - 5 secs panic zone.**

From the experimentation, we collected the following observations:

- There is not a significant difference in offloading performance between deterministic and random seed selection strategies. In effect, from Figure 9**Erreur ! Source du renvoi introuvable.** we cannot spot anymore the existence of long plateau where the dissemination does not evolve. DROID achieves high offloading efficiency by making the re-injection decision dependent not only on the actual dissemination level, but also on the trend of the infection ratio. DROID anticipates and avoids the insurgence of long-lasting plateau in the content diffusion through its reactive strategy. This is not the case in other strategies such as Push & track, and Initial Injection, which pre-calculate the target objective function or the number of initial injections.
- With the best window parameter configurations (3s and 2s) the panic zone ratio is consistently 0%. This means that the Droid re-injection strategy is able to modulate injections very efficiently, without forcing any transmission in the panic zone.
- Performing a sensitivity analysis on the sliding window parameter W, we found that a value of 3s provides the best results. The size of the sliding window trades off how far in time DROID looks back and

dictates the reactivity to sudden changes in the infection ratio. Again, the best value for W is in line with the results that emerged during simulations.

- Globally, DROID shows better results than the previous algorithms across almost any configuration. It represents the best choice, not requiring any previous analysis of the condition of the experiment.

### *4.2.1.3 Comparative analysis across algorithms and configurations.*

This section covers a comparative analysis across the algorithms and configurations that have been tested. Offloading gain is the main metric to maximize. MOTO services are delay tolerant services aiming for the best offloading gain possible, while maximum delay tolerance is not surpassed. In the next graph, a summary of average offloading gains in addition to panic zone ratios are showed.

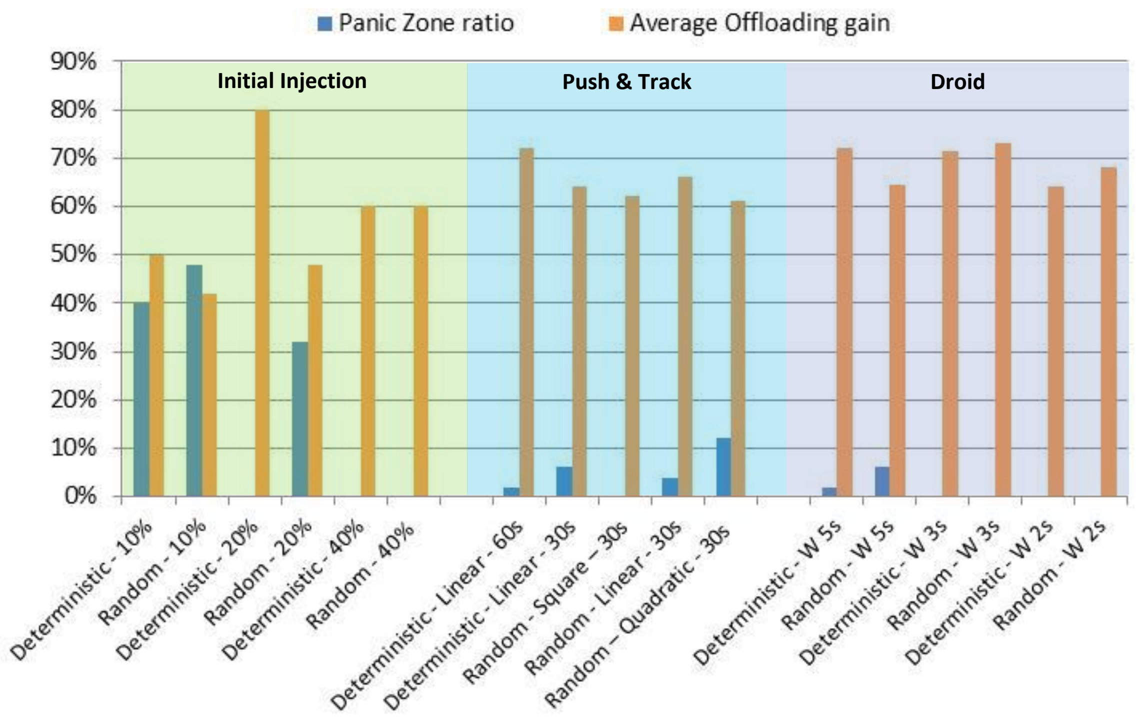

**Figure 10. Average Offloading gain & Panic Zone ratio across algorithms and configurations.**

In order to identify what algorithms and configurations are the best performers, we came up with a coefficient that considers main performance metrics: delay from first injection, offloading gain and battery consumption with different weights (panic zone ratio metric has not been directly considered, since its influence is included in the Offloading Gain). The coefficient is calculated using the following mathematical formula:

Best performing coefficient: 0.15 * (1-Avg.Delay/MAXDelayTolerance) + 0.6*Off.Gain + 0.25*(1-BatteryConsumption/MAXBatteryConsumption)

This formula takes into account the expected performance of the MOTO services and weights the different metrics according to their respective importance, in order to evaluate the different algorithms. In this sense, the offloading gain is weighted the highest, while delay the lowest. Figure 11 represents this coefficient for each of the algorithms and configurations tested.

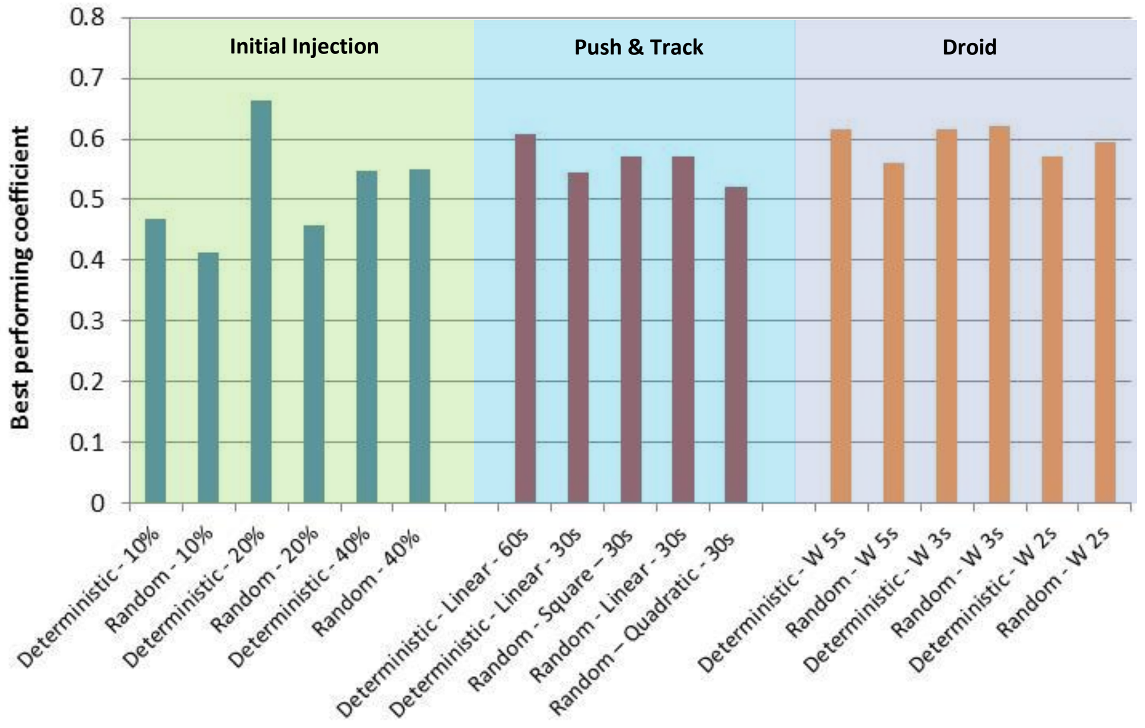

**Figure 11. Best performing coefficient across algorithms and configurations.**

Figure 10 and Figure 11 above shows that DROID algorithm is consistently showing the best performance results, despite "Initial Injection - Deterministic - 20%" is the best performing configuration, under described test conditions. DROID adapts its injection profile to the evolution of the dissemination, in contrast to Push & Track and Initial Injection, which choose in advance the objective function and the number of initial copies, without caring about the dissemination evolution.

In Figure 12 bellow, the behavior in terms of delay tolerance for the best performing configurations for each of the algorithms is shown.

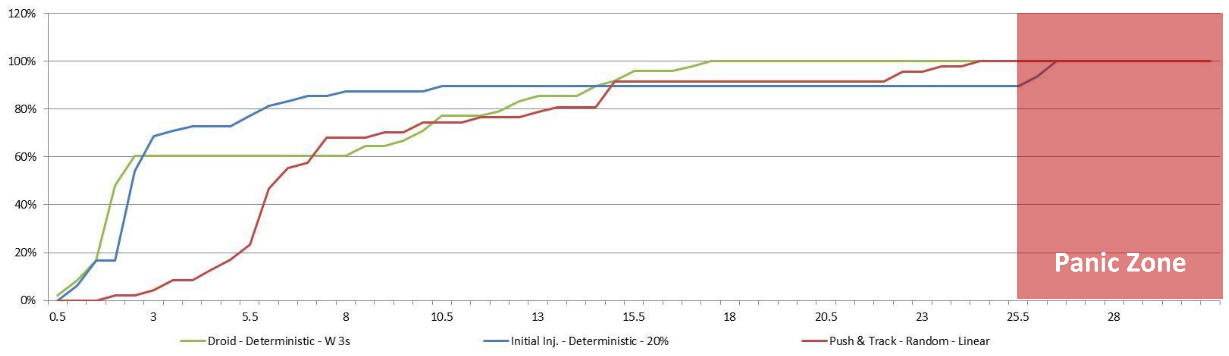

**Figure 12. Dissemination evolution - Best performing configurations per algorithm.**

### 4.2.2 Battery consumption analysis

The main aim of this performance testing was to assess whether duty cycling provides an improvement in terms of battery consumption and at what extend this functionality impacts the performance. The duty cycle was implemented in the prototype through a mobile app which switched on / off Wi-Fi interface as per a configured duty cycle. The following table and graph below depict the performance impact by comparing a reference scenario with duty cycling ON under the same configuration.

**Table 21. Duty Cycle testing. Summary of results.**

| Configuration | | Average Delay from first injection | StdDev Delay from first injection | Panic Zone ratio | Average Offloading gain |
|---|---|---|---|---|---|
| **Initial Injection** | Deterministic - 20% (Ref.) | 5.78 | 7.31 | 0% | 80% |
| | Deterministic - 20% - Duty Cycle | 17.22 | 11.23 | 62% | 25% |
| | Deterministic - 40% (Ref.) | 4.05 | 3.47 | 0% | 60% |
| | Deterministic - 40% - Duty Cycle | 11.97 | 11.85 | 35% | 31% |
| **Push & Track** | Deterministic - Linear - 60s (Ref.) | 17.30 | 18.47 | 2% | 72% |
| | Deterministic - Linear - 60s - Duty Cycle | 23.50 | 16.99 | 2% | 28% |
| | Deterministic - Linear - 30s (Ref.) | 11.92 | 11.03 | 6% | 64% |
| | Deterministic - Linear - 30s - Duty Cycle | 14.28 | 8.45 | 9% | 19% |
| **DROID** | W 2s - 1 Area Ref. | 4.61 | 2.27 | 0% | 83% |
| | W 2s - 1 Area w/ Duty cycle | 10.09 | 5.74 | 3% | 7% |
| | W 2s - 2 Areas Ref. | 4.96 | 3.39 | 0% | 57% |
| | W 2s - 2 Areas w/ Duty cycle | 8.62 | 4.86 | 3% | 13% |
| | W 5s - 2 Areas Ref. | 7.95 | 6.23 | 0% | 58% |
| | W 5s - 2 Areas w/ Duty cycle | 16.82 | 11.97 | 8% | 17% |

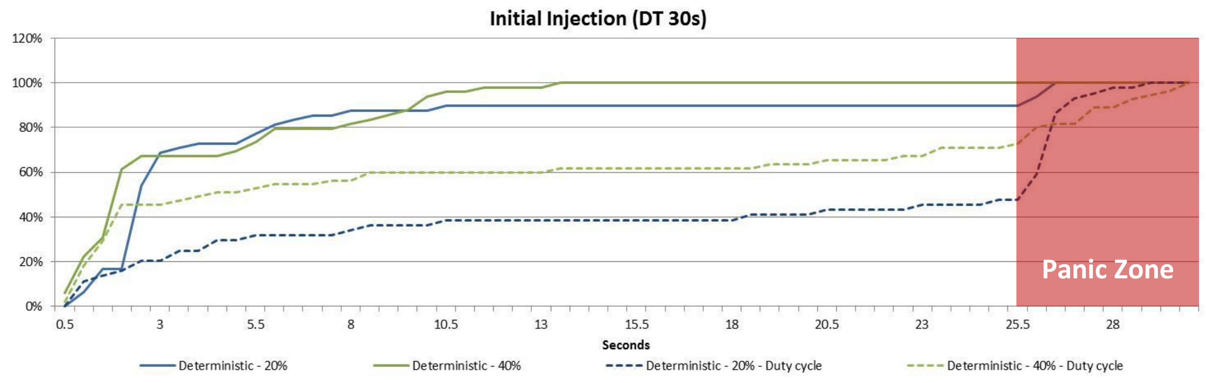

**Figure 13. Dissemination evolution. Initial Injection. Duty cycle testing.**

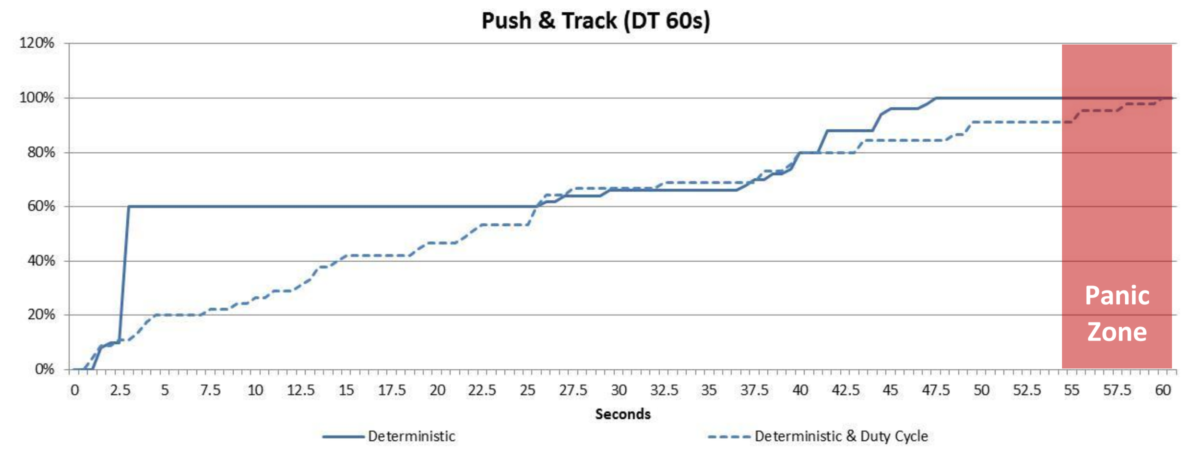

**Figure 14. Dissemination evolution. Push & Track (Delay Tolerance 60s). Duty cycle testing.**

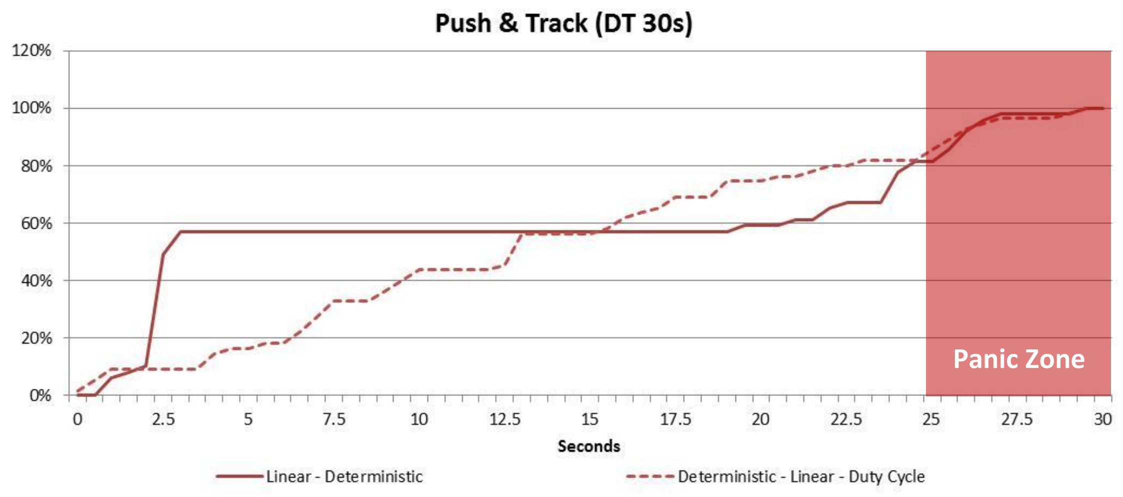

**Figure 15. Dissemination evolution. Push & Track (Delay Tolerance 30s). Duty cycle impact.**

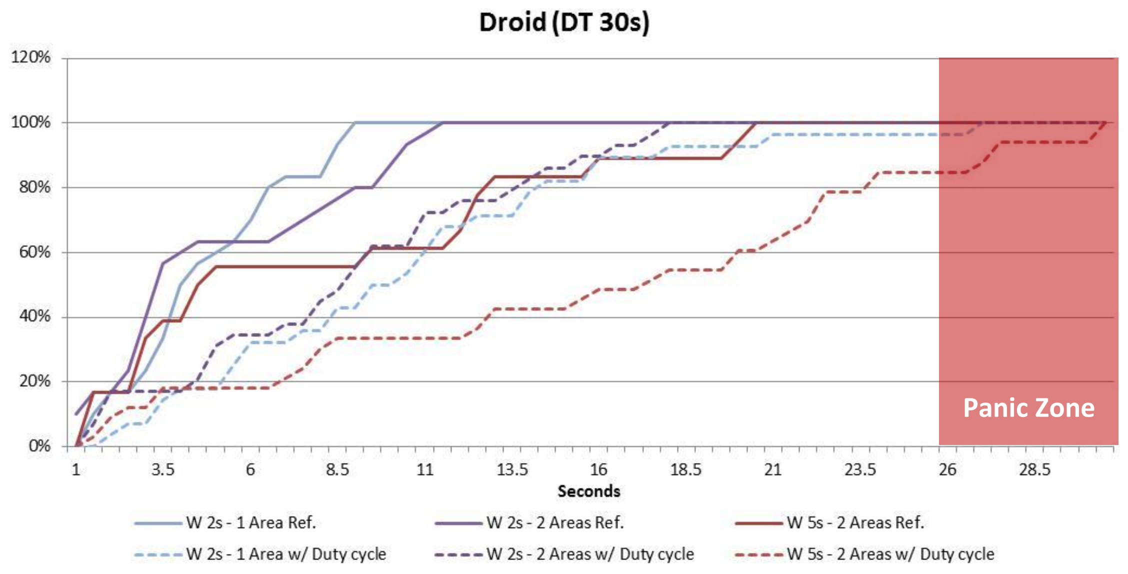

**Figure 16. Dissemination evolution. DROID. Duty Cycle testing.**

Results confirm that duty cycle significantly impacts performance metrics across all the algorithms and different configurations. This is mainly because contact opportunities among users are reduced. For instance, offloading gain results confirm that duty cycling significantly impacts this metric. On the other hand, DROID algorithm showed the best performance taking into account the balance between aspects, battery consumption and offloading gain. Note that this impact was expected based on the theoretical analysis of WP3. As described in [3], when duty cycling is ON, the average end-to-end delay in the opportunistic network is scaled up by a factor approximately equal to $1/\Delta$ where $\Delta$ is the ratio of ON time over the duty cycle period. Therefore, in our experiments, it was expected that the opportunistic dissemination process was much slower when duty cycle is active, resulting in additional injections both before and during the panic zone.

Once this assumption is confirmed, an analysis of the battery consumption is carried out. The amount of battery consumed by the different algorithms was monitored using an external application able to measure battery consumption. Before every test, the remaining battery capacity of the handset was checked and noted together with a timestamp. In order to assess the consumption per hour (mA/h) of each algorithm, when tests for each algorithm finished, the remaining battery capacity and the related timestamp were noted again.

The handset used for this test was a Motorola Moto G running Android 4.4.4 connected to AVEA's LTE network and to FON's Wi-Fi network. In the case of DROID algorithm, battery consumption was assessed during 3rd test fest using commercial 3G / LTE networks. In both cases, the device was in wake up mode (screen on) all the time during the tests to check the correct behavior of the MOTO app. Although this increased the consumption significantly, and probably would not adjust to real conditions, all measurements were done under the same conditions with the objective of ensuring the comparison of each algorithm was fair. Thus, absolute figures should not be taken into account but the comparison among them. For all the tests conducted, the configuration was 20s ON and 20s OFF. However, in practical terms, this meant ~15s ON, ~25s OFF, since it took some time for the handset to activate Wi-Fi interface again.

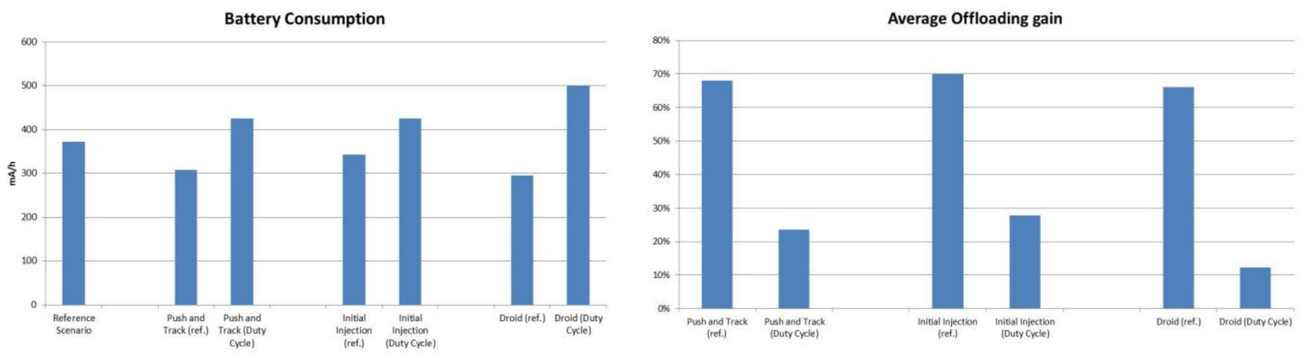

**Figure 17. Battery Consumption analysis.**[5]

In the left chart showed in Figure 17, the battery consumption is compared for each of the algorithms. The "Reference Scenario" bar represents the direct download of the content using the operator's networks (same content, times and users) without the use of any MOTO algorithm. As it can be seen in

[5] Droid algorithm testing on duty cycling and battery consumption where performed during the 3rd test fest with devices connected only to Wi-Fi network. Thus, results are not fully comparable with the ones of other algorithms, since they were collected during the 2nd test fest under different network conditions with devices attached to LTE network. Consumption figures should only be compared with its reference without duty cycling.

Figure 17, results obtained showed that "Push and Track", "Initial Injection" and "DROID" algorithm slightly reduce the battery consumption, being "Push and Track" the best in this aspect.

However, differently from what was expected, all the MOTO algorithms increased handset's battery consumption when Duty Cycle was active (only "Push and Track" and "Initial Injection"). This unexpected behavior was attributed to the fact that Duty Cycle was implemented by switching On/Off the Wi-Fi interface and this process had an additional impact in battery consumption. In a commercial implementation, duty cycling must be implemented in a way that provides energy efficiency benefits.

While the outcome of these tests did not allow us to quantify any energy gain due to duty cycling, results are useful nevertheless. Indeed, they confirm that some specific mechanisms (switching to a low-power mode) should be implemented through features made available directly in the card drivers, and not as patches implemented at the application level. Specifically, the drawbacks of implementing power saving mechanisms at the application level when switching frequency is in the range of our experiments are well-known problems from the general literature on mobile networking, that our experiments just confirm.

As a conclusion, it is worth mentioning that MOTO offloading algorithms demonstrated to consume less battery than the downloading of the content directly through the core networks. Therefore, D2D communications with the designed offloading algorithms show promising results in terms of battery consumption with regards to WAN content download. This is an interesting added value for the end users.

### 4.2.3 Performance impact of security features.

One of the main goals of the 3rd test fest was the assessment of security features impact on MOTO performance. These features were already integrated in the prototype according to described in section 2.6. Reference testing (without security features) was also carried for comparison purposes. This way, the potential delay introduced by the security framework proposed for the MOTO solution was measured. Thus, main metric to assess is the delay from first injection. Security features are only impacting this metric as described in section 2.6. In the graphs and table below, results are summarized.

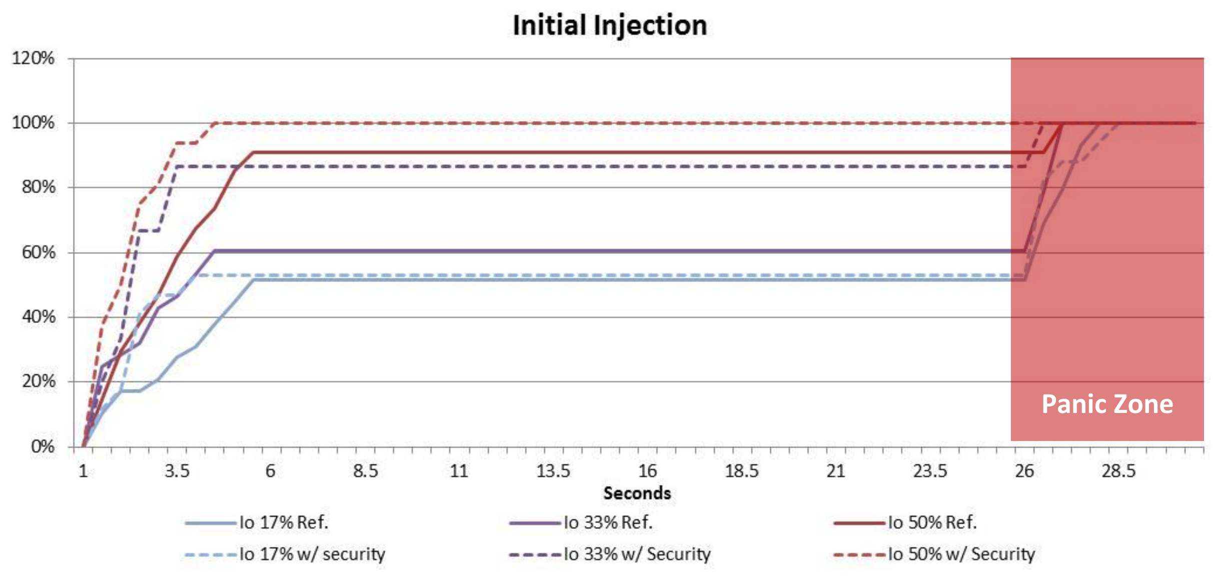

**Figure 18. Dissemination evolution. Initial Injection (DT 30s, PZ 5s). 2 dissemination areas. Security features impact**

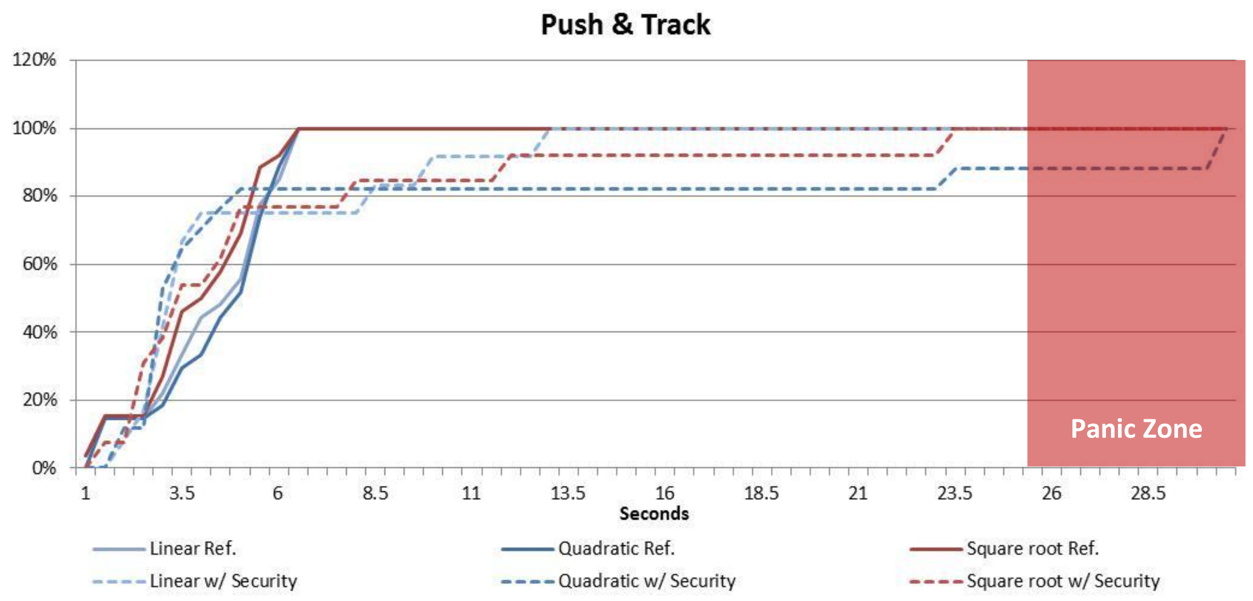

**Figure 19. Dissemination evolution. Push & Track (DT 30s, PZ 5s). A single dissemination area. Security features impact.**

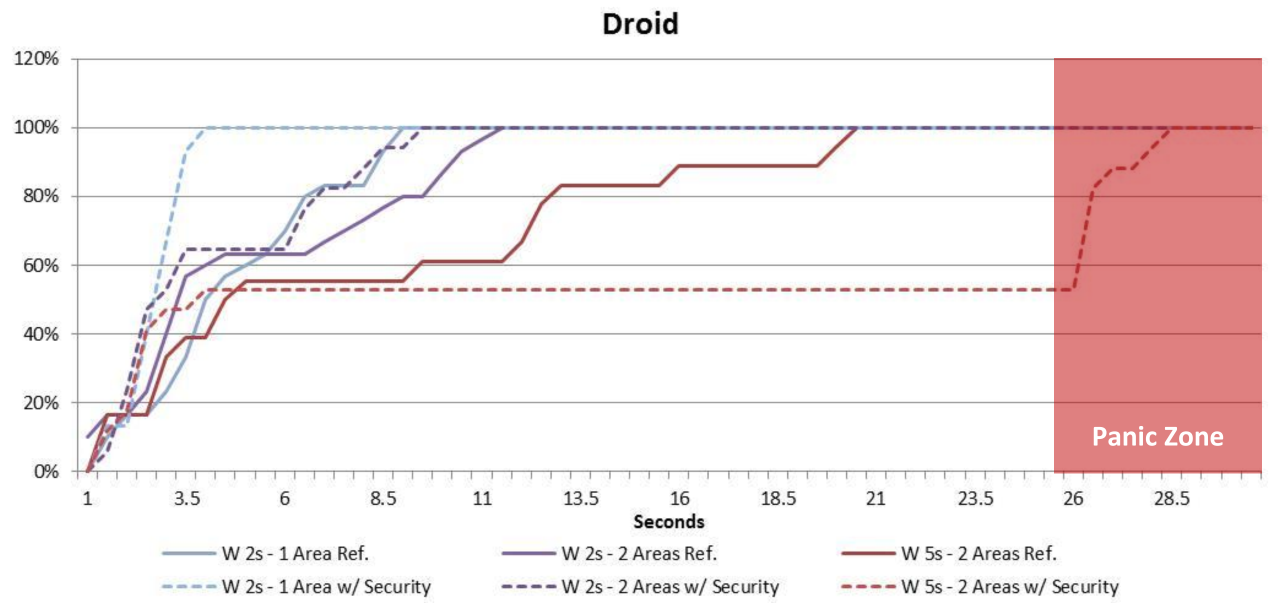

**Figure 20. Dissemination evolution. DROID (DT 30s, PZ 5s). Security features impact.**

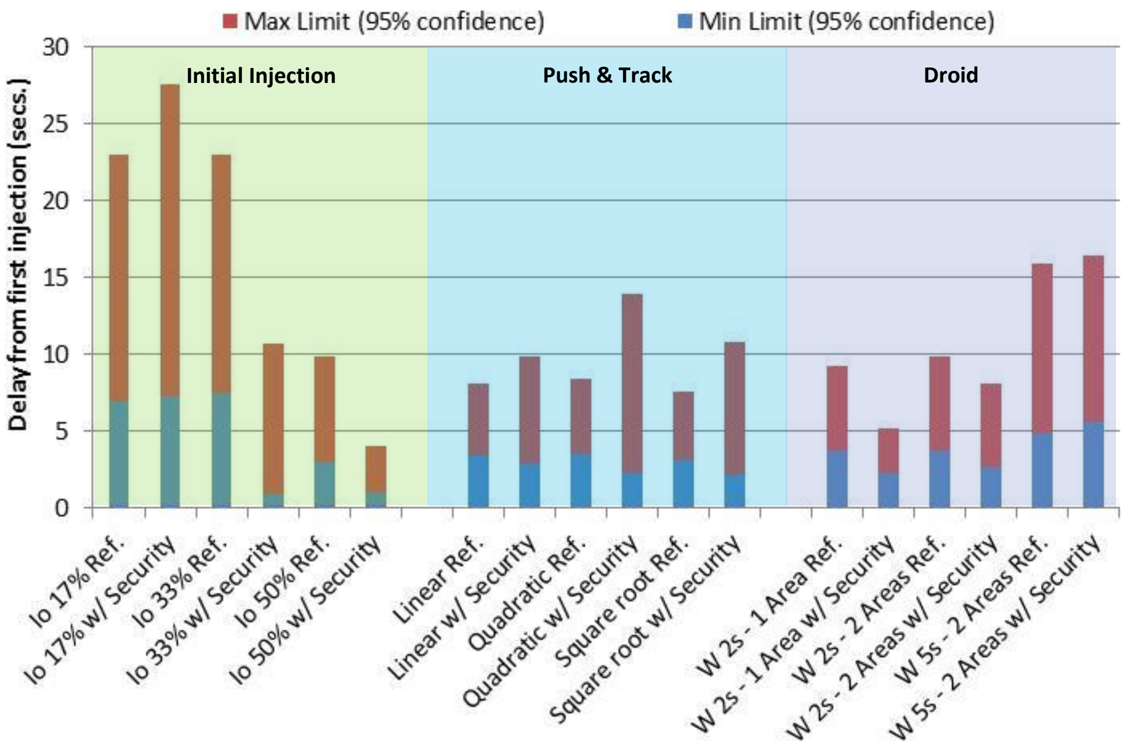

**Figure 21. Delay from first injection: average +- 95% confidence intervals.**

Results show that impact of security mechanisms is so marginal that other factors prevail. The variability that we see across the experiments seems more related to the impact of the background conditions that we do not control, than to the presence or absence of security features. This could be the reason why sometimes security increases delay while some other times it decreases delay. Therefore, we can confirm that security mechanisms do not result in a remarkable degradation of performance, as it was initially expected.

# 5 Conclusions

Performance testing results have allowed us to confirm a bundle of assumptions taken during theoretical analysis and simulations. Additional interesting findings have been also collected across the document, which will allow optimizing future versions of the prototype or eventually a commercial solution. Below, main conclusions are summarized:

- DROID algorithm provided best results across testing scenarios. This confirms that it implements the most adaptive offloading strategy.
- Despite being the simplest algorithm, Initial Injection can provide good results when scenario conditions are well known and slow / non-changing. This algorithm may be chosen when MOTO is used in scenarios like stadiums, which falls in this type of scenarios.
- Information about network conditions and topology can optimize the dissemination strategy, providing better offloading gain results. However, very adaptive algorithms like DROID provide good results even when conditions are un-known. DROID enables the option of delivering a MOTO service quite independent to operator infrastructure, not requiring much integration.
- As expected, services with more relaxed delay requirements will allow higher offloading efficiency. Longer delay tolerance allows exchanging more content on the opportunistic channel.
- MOTO offloading algorithms demonstrated to consume less battery than the downloading of the content directly through cellular networks, showing promising results regarding energy efficiency.
- Duty cycle significantly impacts performance metrics across all the algorithms and different configurations. It was expected that the opportunistic dissemination process was much slower when duty cycle is active, resulting in additional injections both before and during the panic zone
- Differently from what was expected, all the MOTO algorithms increased handset’s battery consumption when Duty Cycle was active. This is attributed to the fact that Duty Cycle was implemented by switching On/Off the Wi-Fi interface and this process had an additional impact in battery consumption. Thus, Duty Cycle should be implemented through features made available directly in the card drivers, and not as patches implemented at the application level
- In the way that security features have been implemented, results show that their impact of is so marginal that other factors related to the testing conditions prevail.

# 6 References

[1] D2.2.2: General architecture of the Mobile Offloading system (release b), MOTO Consortium, August 2015.

[2] D4.2: Protocols for infrastructure offloading control and coordination, MOTO Consortium, October 2014.

[3] D3.2: Spatiotemporal characterization of contact patterns in dynamic networks, MOTO Consortium, April 2014.

[4] D3.3.2: Design and evaluation of enabling techniques for mobile data traffic offloading (release b), MOTO Consortium, March 2015.

## DISCLAIMER