Mobile Opportunistic Traffic Offloading

(MOTO)

D4.1.2: Protocols for terminal-to-terminal communications
Release b

Document information	
Edition	3
Date	28/05/2015
Status	Edition
Editor	UPMC
Contributors	UPMC, TCS, CNR, INNO

Contents

1 Executive summary

This deliverable is the second (and final) release of D4.1 (Protocols for terminal-to-terminal communications). Its main purpose is to report on the project's achievements concerning terminal-to-terminal communication strategies. One of the fundamental design choices of the MOTO project is to follow a *partial coordination* strategy. This means that the influence of the MOTO "offloading coordination agent" is limited to deciding when a content should be injected to a given terminal; the opportunistic domain operates in a stand-alone fashion. Whereas opportunistic communications are in general constrained by limited duration and capacity, users, conversely, generate, consume, and share contents that are becoming increasingly larger. In such a situation, opportunistic content-sharing solutions must be reformulated to support efficient dissemination of large contents. In particular, data must be sliced so that smaller pieces are transmitted separately; this leads to a better use of short-lived contacts and promotes progressive content dissemination. There are a number of challenges that we have to face: (i) upon an encounter, a node must decide which content (when multiple contents must be disseminated) should be given priority (inter-content piece selection), (ii) if the selected content is composed of multiple pieces, the node must choose which piece of the content to send first (intra-content piece selection), (iii) if the contact is long enough to accommodate multiple pieces, how many of them should be sent in a row, (iv) as the initial state of the system has direct impact on the quality of the dissemination (i.e., which nodes should have the first copies of the content), it is also important to determine the initial seeders, and (v) how security solutions operate in the opportunistic domain.

We tackle these challenge with the following complementary contributions. Firstly, we propose EPICS, a protocol designed to quickly exchange large contents in opportunistic networks. EPICS directly addresses the first two challenges. Using grey relational analysis, EPICS is able to balance the distribution of contents that have different sizes and creation times, providing fairer delay distribution and faster dissemination. We evaluated the performance of EPICS in a testbed composed of ten Android smartphones and showed that, when compared to a uniform strategy, EPICS ensures fairer dissemination delays for all the contents regardless of their creation times and sizes. Although EPICS leads to very good results, it does not get the most of longer contacts (that can accommodate several pieces in a row. To address this specific issue, we propose an extended version of EPICS (called DAD) that selects *on-the-fly* the most appropriate number of pieces to transmit in a burst. Through experimental evaluation, we show significant improvements over the baseline solution. DAD has been selected as the final MOTO terminal-to-terminal dissemination protocol. We consider then the problem of identifying the best nodes to inject a content through the infrastructure so that the dissemination on the opportunistic domain evolves properly. We propose an adaptive offloading solution based on the reinforcement learning framework and we evaluate and compare the performance of two well known learning algorithms: Actor Critic and Q Learning. More precisely, in our solution the controller of the dissemination process, once trained, is able to perform at any time the most appropriate choice about the number of content replicas to be injected in the opportunistic network to guarantee the timely delivery of contents to all interested users. We show that our reinforcement learning based system is able to automatically learn the best strategy to reduce the traffic on the cellular network, without relying on any additional context information about the opportunistic network. Finally, our solution reaches higher level of offloading w.r.t. other state of art approaches, in a range of different mobility settings. Finally, we consider security aspects involved

in device-to-device communications. We propose first to investigate integrity-preserving data dissemination in D2D networks and, in particular, the problem of reliable communication in a multihop D2D network despite the presence of malicious attacks. The problem proves difficult since even a single malicious attacker node, if not neutralized, can lie to the entire network. We propose then a security solution for D2D communications that consider integrity, confidentiality, and encryption as the fundamental substrate for the MOTO security framework.

2 Introduction

Concert halls, stadium, bus waiting shelters, are just few examples of places where collocation creates common interests between people. In such locations, it could be useful to locally exchange photos, audio files, travel information or headline news, using what we always have with us: our smartphones. In the latest few years, these devices have witnessed a very quick evolution and a high wide-spreading market penetration. It is expected that traffic from wireless and mobile devices will surpass traffic from wired devices by 2016 [1]. The reason of this success is to ascribe also to both the increasing computational power and the wide range of connectivity interfaces embedded, making them suitable to use in ad hoc and opportunistic networks. All the evident advantages of opportunistic networks (fault tolerance, locality, scalability, infrastructure offloading) can lead to a new widespread content-centric, web 2.0 or media-sharing mobile applications such as proximity chat, local social networks, video and photo sharing, folksonomy, or microblogs.

One of the fundamental design choices of the MOTO project is to follow a *partial coordination* strategy. This means that the influence of the MOTO "offloading coordination agent" is limited to deciding when a content should be injected to a given terminal; the opportunistic domain operates in a stand-alone fashion. This is illustrated in Figure 1, which shows an overview of the MOTO specification architecture.

Whereas opportunistic communications are in general constrained by limited duration and capacity, users, conversely, generate, consume, and share contents that are becoming increasingly larger. In such a situation, opportunistic content-sharing solutions must be reformulated to support efficient dissemination of large contents. In particular, data must be sliced so that smaller pieces are transmitted separately; this leads to a better use of short-lived contacts and promotes progressive content dissemination. There are a number of challenges that we have to face:

- **Challenge 1.** Upon an encounter, a node must decide which content (when multiple contents must be disseminated) should be given priority (inter-content piece selection).
- **Challenge 2.** If the selected content is composed of multiple pieces, the node must choose which piece of the content to send first (intra-content piece selection).
- **Challenge 3.** As the initial state of the system has direct impact on the quality of the dissemination (i.e., which nodes should have the first copies of the content), it is also important to determine the initial seeders.
- **Challenge 4.** Security is a key aspect of the MOTO project. In this deliverable, we describe the security aspects considered in the project, as well as the proposed solutions with regard to device-to-device communications.

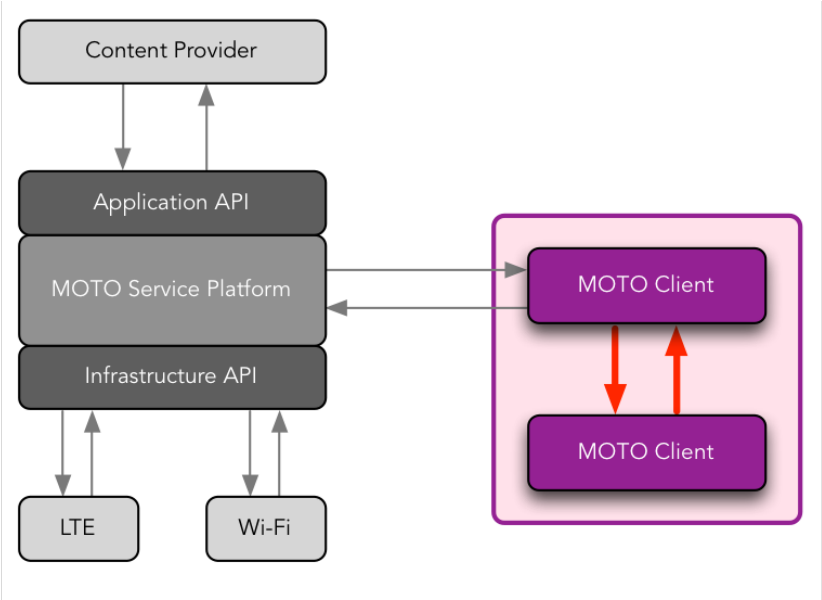


Figure 1: MOTO specification overview.

To address the first two challenges, we propose EPICS, a distributed protocol to help nodes decide which is the best piece to transmit during a contact in order to achieve a predefined dissemination policy. In a nutshell, our goal is to design an efficient strategy for deciding *which piece to transmit and from which content* whenever a communication opportunity happens. EPICS chops contents into small pieces and shuffle them to speed up the dissemination. To decide which piece from which content should be sent, EPICS relies on the grey relational analysis with the goal to prioritize contents having some expected features (most popular, most recent or most urgent content). EPICS follows a "prevalence" principle meaning that nodes try to first send pieces that are rarer in the network, based on the node's own view of the dissemination process.

We evaluate the performance of EPICS using an Android application that enables the dissemination of media content between collocated Android devices in ad hoc mode. We set up a testbed composed of 10 Android phones. We first assess the enhancement of a better intra-content selection strategy. For the sake of completeness, we evaluate the behavior of EPICS in terms of both intra-content and inter-content piece selection. We compare the results with a strategy in which content is fully sent at each encounter (namely full content spreading – FCS). Second, we evaluate the inter-content piece selection strategy. In this case, we compare EPICS with a strategy where the content to exchange is chosen in an uniform random way among all the available contents. We first perform the experimentation in a static scenario in order to ensure the same configuration for both strategies. Third, we introduce node mobility in the scenario by creating an emulated mobility and by performing a real mobility scenario. Results obtained in both static and mobile scenarios show that, in practice, EPICS leads to higher dissemination ratios and shorter dissemination delays.

We consider the problem of determining the initial seeders (challenge 3) in a separate section, as it is not directly related to the terminal-to-terminal communication part. In particular, we argue that dissemination protocols can benefit from a learning mechanism to derive the most ap-

propriate re-injection strategy. Among the many existing RL techniques, we used and evaluated two well known and very robust solutions: the Actor Critic and the Q-Learning algorithms. For nodes selection we adopted a heuristic mechanism that permits to identify, online, what nodes are more useful to spread contents. We propose an adaptive offloading solution based on the reinforcement learning framework and we evaluate and compare the performance of two well known learning algorithms: Actor Critic and Q Learning. In our solution, the controller of the dissemination process, once trained, is able to perform at any time the most appropriate choice about the number of content replicas to be injected in the opportunistic network to guarantee the timely delivery of contents to all interested users. We show that our reinforcement learning based system is able to automatically learn the best strategy to reduce the traffic on the cellular network, without relying on any additional context information about the opportunistic network.

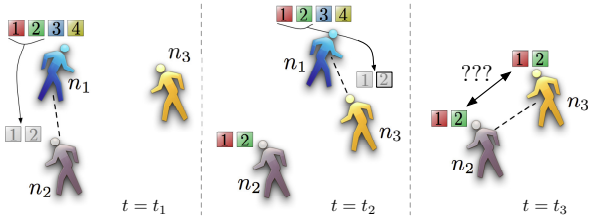
With regard to security (challenge 4), we tackle it through two complementary viewpoints. Firstly, we investigate the problem of reliable communication in a multihop D2D network despite the presence of malicious attacks. The problem proves difficult since even a single malicious attacker node, if not neutralized, can lie to the entire network. We prove the necessary and sufficient condition for reliable communication in dynamic networks, in the presence of up to k malicious attackers. We consider the two cases where cryptography is available and not available. We apply then these conditions to two case studies: participants interacting in a conference, and agents moving in the Paris subway. Secondly, we propose the MOTO security protocol for terminal-to-terminal communications. In particular, the proposal allows the secure distribution of pseudonyms and correlated encryption key pairs, content distribution, and feedback delivery. The protocol has been analyzed under different attacks, such as man in the middle, and it has until now demonstrated to be robust against malicious attempts against content confidentiality and integrity and user identity disclosure.

The remainder of this deliverable is structured as follows. In Section 3, we present the design and evaluation of EPICS. In Section 5, we address the question of determining good candidates to receive copies of a content from the infrastructure. We address security issues and present MOTO solutions in Section 6. We conclude this document by providing a summary of our contributions as well as some research directions for future developments to be reported in the updated version of this deliverable (D4.1.2, due M32).

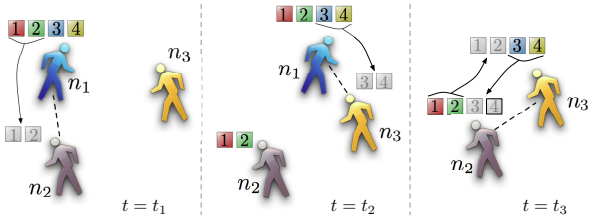
3 EPICS: Terminal-to-terminal dissemination with intra- and inter-content piece selection

Opportunistic content-sharing solutions must be reformulated to support efficient dissemination of large contents. In particular, data must be sliced so that smaller pieces are transmitted separately; this leads to a better use of short-lived contacts and promotes progressive content dissemination. The first challenge is then to choose which piece of the content to send upon a contact. The problem becomes even more challenging when multiple contents flow in the network at the same time. In a nutshell, the goal of this work is to design an efficient strategy for deciding *which piece to transmit and from which content* whenever a communication opportunity happens.[1]

[1] N. Belblidia, M. Sammarco, L. H. M. K. Costa, and M. Dias de Amorim, "EPICS: Fair Opportunistic Multi-Content Dissemination", to appear in *IEEE Transactions on Mobile Computing*, 2015.



(a) Sequential.



(b) Random.

Figure 2: A motivating example. Selecting the pieces to transfer is fundamental to efficient dissemination of fragmented contents.

3.1 Preliminaries

In this section, we define the problem of disseminating multiple contents over opportunistic networks. We provide all the necessary background before introducing the dissemination protocol. In our problem, a relatively large *content* must be disseminated to a population of mobile nodes that communicate in an opportunistic fashion. To reduce the dissemination delay, the content is sliced into a number of *pieces* of equal size, which allows benefiting from shorter contacts than the one necessary to transfer the entire content.

3.1.1 Piece selection: A motivating example

We now illustrate why the proper selection of pieces to send is important. The straightforward approach for a node to disseminate content in an opportunistic network is to transfer pieces based on an increasing order of identifiers. We will call this strategy *sequential* in the remainder of this document.

We show in Figure 2(a) the sequential approach at three consecutive time instants. In the very beginning, only node n_1 has the content (composed of four pieces). At $t = t_1$, n_1 meets n_2. This latter has no pieces yet. The contact allowing the transfer of two pieces, n_1 sends then pieces 1 and 2. At $t = t_2$, n_1 meets n_3 (which does not have any pieces either). As for the previous case, n_1 transfers the first two pieces. At $t = t_3$, node n_1 has left the network. When n_2 and n_2 meet, the contact opportunity cannot be used because both nodes have the same pieces (in fact, such a situation happens quite frequently in practice).

The ideal case would have been the one in Figure 2(b). Node n_1, instead of disseminating the same pieces each time it meets a node, applies some randomized strategy to avoid the situation described above. Here, at $t = t_3$, nodes n_2 and n_3 are able to exchange pieces turning the encounter into a useful contact.

In a real network composed of dozens or even hundreds of nodes, contact patterns are expected to be much more complex than the example above. It is fundamental then to find a proper piece selection strategy so that it is worth offloading data to the opportunistic domain. Note that the problem becomes even more challenging when multiple content circulate in the network.

3.1.2 Problem definition

Suppose a set of relatively large contents that must be opportunistically disseminated through a population of mobile nodes. In order to speed up the dissemination, contents are sliced into pieces of equal size that are sent separately; we consider a unique piece size to avoid giving priority to pieces besides the priority related to the prevalence of the pieces in the network. Nodes locally store an availability vector per content. The availability vector keeps track of the content pieces that the node holds. When a communication opportunity happens, each node selects a single content and sends the corresponding availability vector to one of its neighbors in order to get missing pieces. The neighbor tries then to send back missing content pieces. We call the method used to select, among available contents, the one to be considered during a contact the *inter-content selection strategy*. Similarly, we call the method used to select, among available pieces, the one to be transferred during a contact the *intra-content selection strategy*.

3.1.3 Network model and assumptions

Let $\mathbf{N} = \{n_0, n_1, \ldots, n_{N-1}\}$ be the set of N mobile nodes in the network. We do not assume any knowledge of mobility patterns. We assume that all nodes in the network are interested in a set of contents $\mathbf{C} = \{c_0, c_2, \ldots, c_{C-1}\}$. Each content c_j is initially only available at a single data source. We do not make any assumption on the creation time of contents.

For each content c_j, the data source chops the content into K_j pieces of equal size (the piece size can be accurately determined to optimize communication opportunities [2]). Contents could have different sizes, hence, the number of pieces depends on the content size. Pieces are sequentially identified as $c_j = \{d_0, d_1, \ldots, d_{K_j-1}\}$. Nodes use their contact opportunities to get pieces, i.e., we assume that there is no infrastructure to help the dissemination process. Nodes can get pieces from the data source and from any other node in the network having it.

Each node n_i locally stores an *availability bitmap vector* $\mathbf{a}_{n_i,j} = \{a_0, \ldots, a_{K_j-1}\}$ and a *prevalence vector* $\mathbf{p}_{n_i,j} = \{p_0, \ldots, p_{K_j-1}\}$ both associated with every known content c_j. The availability bitmap vector $\mathbf{a}_{n_i,j}$ keeps track of c_j content pieces that the node n_i holds. It contains binary values associated to each piece, where $a_m = 1$ if the node n_i has piece d_m, and $a_m = 0$ otherwise. The goal of the prevalence vector is to give a local view of the prevalent pieces in the network. Initially, each node associates an empty prevalence vector to each content at each node. We will see in Section 3.2 how these vectors are updated.

All the variables are summarized in Table 1.

3.2 Intra-content piece selection

In order for the reviewer to have enough material to follow the remainder of the section, let us first present the proposed solution for the *intra-content piece selection* problem. The goals of the solution are to achieve fast content dissemination while keeping the overhead low. To this

Table 1: Summary of variables.

Variable	*Definition*
$\mathbf{N}$	Set of nodes in the network
N	Number of nodes in $\mathbf{N}$
$\mathbf{C}$	Set of contents to be disseminated
C	Number of contents in $\mathbf{C}$
K_j	Number of pieces that compose content c_j
$\mathbf{a}_{n_i,j}$	Availability bitmap associated to content c_j at node n_i
$\mathbf{p}_{n_i,j}$	Prevalence vector associated to content c_j at node n_i

Algorithm 1 n_i Intra-content piece selection strategy

1: **while** $contact\_with(n_j)$ **do**
2: $\quad receive\_from(n_j, \mathbf{a}_{n_j,0})$;
3: $\quad \mathbf{p}_{n_i,0} \leftarrow \mathbf{p}_{n_i,0} + \mathbf{a}_{n_j,0}$;
4: $\quad$ **if** $(\mathbf{a}_{n_i,0} \wedge (\neg \mathbf{a}_{n_j,0}) \neq \emptyset)$ **and** $(initiate\_connection\_with(n_j))$ **then**
5: $\quad\quad d_{s_{i\rightarrow j}} \leftarrow prevalence\_selection\_from((\mathbf{a}_{n_i,0} \wedge (\neg \mathbf{a}_{n_j,0})), \mathbf{p}_{n_i,0})$;
6: $\quad\quad send\_to(n_j, d_{s_{i\rightarrow j}})$;
7: $\quad$ **end if**
8: $\quad$ **if** $(\mathbf{a}_{n_j,0} \wedge (\neg \mathbf{a}_{n_i,0}) \neq \emptyset)$ **and** $(connection\_initiated\_by(n_j))$ **then**
9: $\quad\quad receive\_from(n_j, d_{s_{j\rightarrow i}})$;
10: $\quad\quad \mathbf{i}_{d_{j\rightarrow i}} \leftarrow \{i_0, \ldots, i_{K_0-1}\}$; $i_k = 0$, $\forall k < K_0$ $(k \neq s_{j\rightarrow i})$, $i_{s_{j\rightarrow i}} = 1$
11: $\quad\quad \mathbf{a}_{n_i,0} \leftarrow \mathbf{a}_{n_i,0} \vee \mathbf{i}_{d_{j\rightarrow i}}$;
12: $\quad$ **end if**
13: **end while**

end, nodes keep track of the dissemination progress of each piece, so that they can appropriately prioritize their transmissions without inducing additional communication overhead.

Let us assume, for the time being, that only a single content c_0 is available in the network. We consider the multi-content case later on.

Initially, all nodes in $\mathbf{N}$, except the one where c_0 was produced, have neither prevalence nor availability vectors associated to content c_0 because they are not aware of the presence of c_0 in the network. Nodes create these vectors as soon as they receive the availability vector relative to the new content c_0, $\mathbf{a}_{n_i,0}$, from a node n_i. When nodes n_i and n_j meet, they exchange their availability vectors $\mathbf{a}_{n_i,0}$ and $\mathbf{a}_{n_j,0}$. Node n_i (resp. n_j) computes $\mathbf{a}_{n_i,0} \wedge (\neg \mathbf{a}_{n_j,0})$ (resp. $\mathbf{a}_{n_j,0} \wedge (\neg \mathbf{a}_{n_i,0})$), which gives the candidate pieces to be transferred. They also update their prevalence vectors respectively as: $\mathbf{p}_{n_i,0} \leftarrow \mathbf{p}_{n_i,0} + \mathbf{a}_{n_j,0}$, and $\mathbf{p}_{n_j,0} \leftarrow \mathbf{p}_{n_j,0} + \mathbf{a}_{n_i,0}$. Among the candidate pieces to be transferred, nodes select the one with the lowest prevalence. In the case of a tie, a piece is chosen in a uniformly distributed random way. Let $d_{s_{i\rightarrow j}}$ be the piece sent by n_i to n_j and $d_{s_{j\rightarrow i}}$ be the piece sent by n_j to n_i. After one round of exchanges, nodes update their availability vectors as: $\mathbf{a}_{n_i,0} \leftarrow \mathbf{a}_{n_i,0} \vee \mathbf{i}_{d_{s_{j\rightarrow i}}}$, and $\mathbf{a}_{n_j,0} \leftarrow \mathbf{a}_{n_j,0} \vee \mathbf{i}_{d_{s_{i\rightarrow j}}}$, where $\mathbf{i}_{d_{s_{i\rightarrow j}}}$ and $\mathbf{i}_{d_{s_{j\rightarrow i}}}$ are K_0 element vectors with all positions set to 0 except the position relative to the piece just received, which is set to 1. Note that prevalence vectors have a limited influence at the beginning, but they gain importance as nodes move and exchange pieces. The steps achieved by node n_i are stated in Algorithm 1.

3.3 Inter-content piece selection

In this section, we review the concept of grey relational analysis and illustrate with an example how it is used in EPICS.

3.3.1 Grey relational analysis

Grey system theory is employed in the study of non-deterministic systems. Unlike traditional probability and statistics techniques that are based on large size samples and on assumptions about the distributions of these samples, the grey system theory was developed to study problems of "small samples and poor information" [3, 4]. A situation where no information is available is called "black". Conversely, a situation with complete information is called "white". Grey systems handle intermediate situations.

In this work, we are interested in the grey relational analysis (GRA), which is a method in grey system theory for analyzing discrete data series. Based on the grey relational grade, this method enables measuring the degree of approximation of given data series x_u according to the reference data series x_0.

In the following, we summarize the steps needed for grey relational analysis processing:

(a) Set up the reference data series x_0

$$x_0 = \{x_0(1), x_0(2), \ldots, x_0(M)\},$$

where M is the number of considered data (or metrics) in the analysis. $x_0(v)$ represents the most favored value of the v^{th} data ($1 \leq v \leq M$).

(b) Define the comparison data series x_u

$$x_u = \{x_u(1), x_u(2), \ldots, x_u(M)\},$$

where $1 \leq u \leq S$ and S is the number of compared data series in the analysis.

(c) Compute the difference data series Δ_u

$$\Delta_u = \{\Delta_u(1), \Delta_u(2), \ldots, \Delta_u(M)\},$$

where $\Delta_u(v) = |x_0(v) - x_u(v)|$.

(d) Get the global maximum value Δ_{max} and the global minimum value Δ_{min} from all data series

$$\Delta_{max} = \max_u(\max_v \ \Delta_u(v)),$$
$$\Delta_{min} = \min_u(\min_v \ \Delta_u(v)).$$

(e) Obtain, for each data v in each data series u, the grey relational coefficient $\gamma_u(v)$

$$\gamma_u(v) = \frac{\Delta_{min}+\varsigma\Delta_{max}}{\Delta_u(v)+\varsigma\Delta_{max}},$$

where ς is a coefficient value between 0 and 1. ς is used to compensate the effect of Δ_{max}. Generally, ς is set to 0.5.

(f) Compute the grey relational grade for each data series u

$$\Gamma_u = \sum_{v=1}^{M}(\gamma_u(v) \times w(v)),$$

where $w(v)$ is the weight of the v^{th} data in each series (with, $\sum_{v=1}^{M} w(v) = 1$). If all data in series have the same weight, Γ_u becomes:

$$\Gamma_u = \frac{1}{M}\sum_{v=1}^{M}\gamma_u(v).$$

As mentioned above, the grey relational grade value Γ represents the degree of approximation to the reference data series x_0. A high Γ_u indicates that the values in the data series x_i are, in general, close to the most favored values.

(g) Sort the k values of Γ into descending order.

3.3.2 Principles of the protocol

We use the grey relational analysis to weight the random selection of the availability vector to transmit at each encounter. In our case study, the goal is to ensure a fairer dissemination delay for all contents regardless their creation times and their sizes. To reach this objective, weights should take into account both freshness and content size. Therefore, we consider these two metrics in the grey relational analysis ($M = 2$). Because we set the piece size to a fixed value, content size is defined as the number of pieces. Freshness is defined as the creation time, i.e., the time at which the data source generates a given content. As mentioned in the introduction, more complicated dissemination policies could be applied by defining the appropriate metrics of the grey relational analysis. To illustrate the extensibility of this technique, we could also consider, for example, a metric called content popularity to the x_0 data series, thus differentiating content exchange services to give priority to more popular contents or contents associated with different delay tolerances. Our strategy is suitable in the multi-group-multi-topic generic case too. Announcements of contents belonging to groups or topics a user is not subscribed to, will be immediately filtered out at the reception. For the others, the x_0 data series might be extended to accommodate values related to groups and topics reflecting user's preferences.

In the following, we detail how EPICS operates in our case study. To get the same scale for both metrics, every node first normalizes the evaluated values. To this end, each node gets the current values of both metrics associated to all known contents, takes the maximum and minimum values, and rescales the values in the range $[0, 1]$. Next, every node defines the reference data series. Since contents that are fresher and/or larger take more time to be disseminated,

Algorithm 2 n_i EPICS

$x_0 = \{1, 1\}$
$\Delta_{max} = 1$;
$\Delta_{min} = 0$;
while $contact\_with(n_j)$ **do**
  if not sending() **and not** receiving() **then**
    for u **in** $range(known\_contents())$ **do**
      $x_u \leftarrow get\_normalized\_freshness\_and\_size(c_u)$;
      $\Delta_u \leftarrow compute\_difference\_data\_series(x_0, x_u)$;
      $\Gamma_u \leftarrow compute\_grey\_grade(\Delta_u, \Delta_{max}, \Delta_{min})$;
    end for
    $prob \leftarrow \{\frac{\Gamma_1}{\sum_u \Gamma_u}, \ldots, \frac{\Gamma_k}{\sum_u \Gamma_u}\}$,
$k = range(known\_contents())$
    $s \leftarrow weighted\_random\_selection\_of\_content(prob)$
    $send\_to(n_j, \mathbf{a}_{n_i,s})$;
  end if
end while

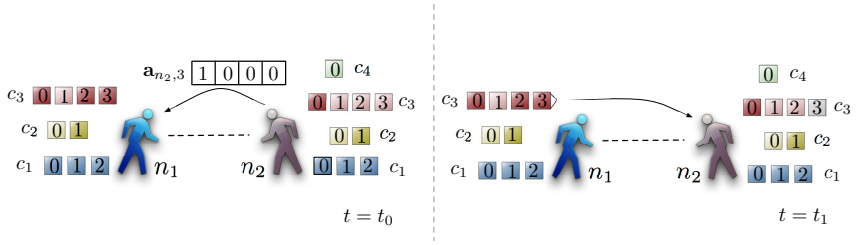


Figure 3: An illustrative example of EPICS. Initially, n_1 has content pieces $c_1 = \{d_0, d_1, d_2\}$, $c_2 = \{d_1\}$, and $c_3 = \{d_0, d_2, d_3\}$. n_2 has content pieces $c_1 = \{d_0, d_1, d_2\}$, $c_2 = \{d_1\}$, $c_3 = \{d_0\}$, and $c_4 = \{d_0\}$.

they should get higher weights. Hence, x_0 is set to $\{1, 1\}$. Then, each node computes Γ values of contents. We assigned the same weight to both metrics ($w(1) = w(2) = \frac{1}{2}$) and set ς to 0.5. Based on Γ values, weights are assigned to each content. The weights are then used to define the probability of selection of the corresponding availability vector. Algorithm 2 details the content selection strategy applied at each node. Note that, since x_u values are normalized, Δ_{max} is always equal to 1 and Δ_{min} is always equal to 0.

3.3.3 Example

We illustrate in Figure 3 how EPICS operates. At $t = t_0$, n_1 knows three contents $\{c_1, c_2, c_3\}$ and n_2 knows four contents $\{c_1, c_2, c_3, c_4\}$. Suppose that contents c_1, c_2, c_3, c_4 were created at times 10, 30, 50, and 60 and have a size of 3, 2, 4, and 1 piece(s) respectively. Before sending one of its availability vectors, n_2 performs the following steps. First, it sets up x_u data series associated to every content c_u. To this end, n_2 gets normalized values for both freshness and size metrics. As previously mentioned, freshness is defined as the time of content creation. Thus, freshness values are $\{10, 30, 50, 60\}$. After normalization, they become $\{0, 0.4, 0.8, 1\}$. Size is defined as the

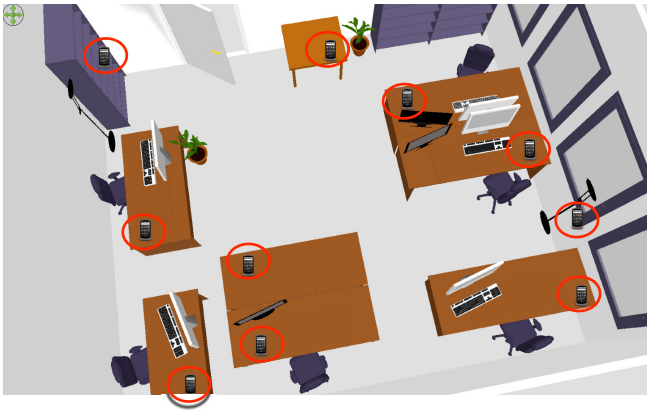


Figure 4: Configuration of the experiment. We used ten smartphones running Android 2.3.3 and placed them at fixed locations in an office of the university.

number of pieces. Hence, sizes are $\{3, 2, 4, 1\}$. After normalization, sizes equals $\{0.66, 0.33, 1, 0\}$. Hence, the data series are: $x_1 = \{0, 0.66\}$, $x_2 = \{0.4, 0.33\}$, $x_3 = \{0.8, 1\}$, and $x_4 = \{1, 0\}$, with data series x_u associated to content c_u. Second, n_2 computes the difference data series Δ_u according to the reference series $x_0 = \{1, 1\}$: $\Delta_1 = \{1, 0.33\}$, $\Delta_2 = \{0.6, 0.66\}$, $\Delta_3 = \{0.2, 0\}$, and $\Delta_4 = \{0, 1\}$. Third, n_2 sets $\Delta_{max} = 1$ and $\Delta_{min} = 0$ and obtains the grey relational coefficients: $\gamma_1 = \{0.33, 0.6\}$, $\gamma_2 = \{0.45, 0.43\}$, $\gamma_3 = \{0.71, 1\}$, and $\gamma_4 = \{1, 0.33\}$. Fourth, n_2 determines the grey relational grades: $\Gamma_1 = 0.46$, $\Gamma_2 = 0.44$, $\Gamma_3 = 0.85$, and $\Gamma_4 = 0.66$. Finally, n_2 assigns content selection probabilities: $prob = \{0.19, 0.18, 0.35, 0.27\}$. Then, one content is selected based on these probabilities. Suppose that content c_3 is chosen. Accordingly, n_2 sends c_3's availability vector $\mathbf{a}_{n_2,3}$ to n_1. At time $t = t_1$, among c_3 candidate pieces to be transferred, n_1 selects the one with the lowest prevalence (assume it is d_3) and sends it to n_2.

3.4 Experimental evaluation setup

We have adopted an experimental approach to evaluate EPICS. In this section, we summarize our experimental setup.

3.4.1 Experimental parameters

Transport protocol. Data pieces are sent using UDP datagrams (reliable TCP connections are available too).

Number of nodes. We placed 10 Android phones (4 HTC Desire and 6 Samsung Galaxy-S-II equipped with Android 2.3.3) on fixed locations in a 30m^2 office (Figure 4).

Number of contents. We consider the dissemination of 40 contents. Each device has four original contents stored on their SDCARD. We enabled the batch mode to automate the exchanging process at the starting phase of the experiments.

Piece size. Since pieces are sent through UDP, we set the piece size to fill the maximum UDP packet size (64kB). This way, each piece is transmitted in a unique UDP packet.

Table 2: Parameters of the experiment.

Parameters	*Values*
Transport protocol	UDP
Number of nodes	10
Number of contents	40
Piece size	64kB
Size of contents	16kB, 3.5MB
T_c	1 second
T	2 seconds
m	6

Size of contents. Even if a content may be smaller than the piece size defined above, still a data message must be created and sent through an UDP packet. Thus, we consider that content sizes vary from 1 piece (16kB) to 56 pieces (3.5MB).

Content creation times. Contents are created at different times of the experiment after a warmup period of 120 seconds. For complete automation and to make all the applications start at the same time, we developed an NTP client application to synchronize the smartphone internal clock to an NTP server and we used TaskBomb [5], an application which acts as Unix's Cron utility for Android.

Beaconing parameters. In the following experiments, we want each node always connected to each other. Thus, for the neighborhood management, we set $T_c = 1$ s, $T = 2$ s, and $m = 6$.

Availability vector message parameters. Availability vector messages are broadcasted at the same frequency as beacon messages.

Experiment parameters are summarized in Table 2. Although we tested and validated the working of EPICS protocol in a mobile environment, we first performed the experiment in a static configuration to be able to compare the results obtained using different strategies.

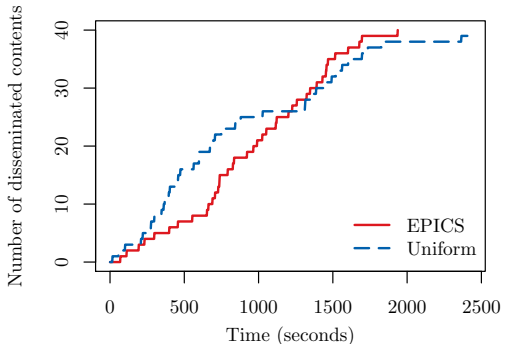
3.4.2 Benchmarking

We compare EPICS with a uniform strategy. We call the uniform strategy, a strategy that selects the content to transmit in a random uniform way. Experiments last for 10 minutes. Content creation times vary between 0 seconds and 1,030 seconds after the warmup period. The experiments last as long as needed for both EPICS and uniform strategies to achieve full dissemination of all pieces.

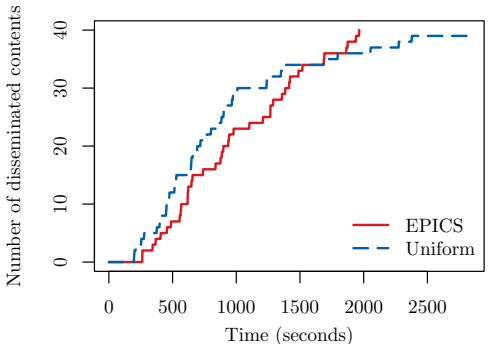
3.5 Experimental results: EPICS versus uniform

In this section, we evaluate the impact of the intra and inter-content selection strategies using our experimental testbed. The objective of our study is to mitigate the variability of content dissemination delays observed when using uniform random inter-content selection. We compare EPICS to a uniform random selection of the availability vector to transmit. As previously stated, we called this strategy *uniform* selection.

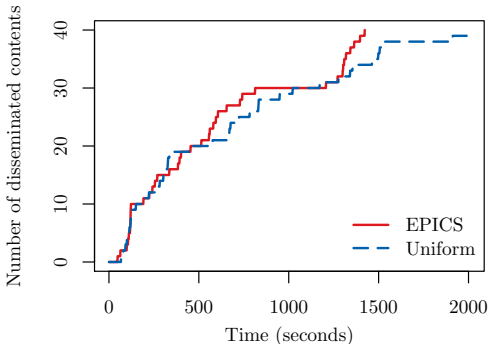
Evolution of the dissemination delay. We vary both content creation times and content sizes to investigate content dissemination evolution. Figure 5(a) shows the number of disseminated



(a) Contents have variable size and variable creation time.



(b) Contents are created simultaneously at time 0 after the warmup period.



(c) Contents have the same size (140kB).

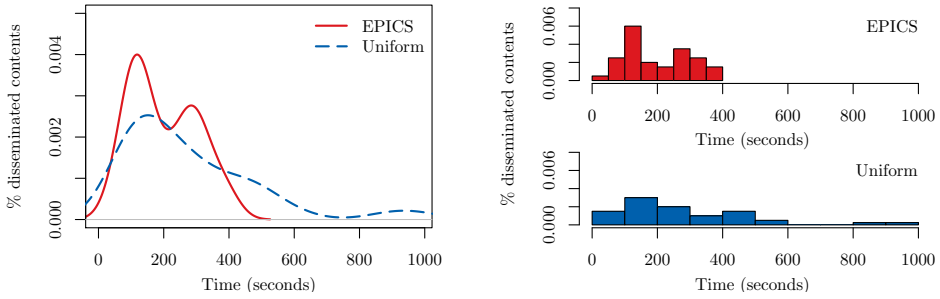
Figure 5: Content dissemination evolution. Although the uniform random selection performs better than EPICS in the beginning, EPICS achieves a faster dissemination of all contents at the end.

contents over time. The uniform selection allows faster dissemination of the first contents. Indeed, the first 25 contents are fully disseminated after 15 minutes (880 seconds) with the uniform strategy, whereas EPICS achieves the same ratio after 19 minutes (1,122 seconds). Since uniform selection keeps selecting the content to transmit in a random uniform way through time, firstly created contents get more chances to be selected compared to EPICS. As a consequence, the dissemination of these first contents is faster. Nevertheless, the uniform selection spends much more time to perform full dissemination of the remaining 15 contents. All contents are fully disseminated after 40 minutes (2,429 seconds), while EPICS finishes the dissemination in 32 minutes (1,937 seconds).

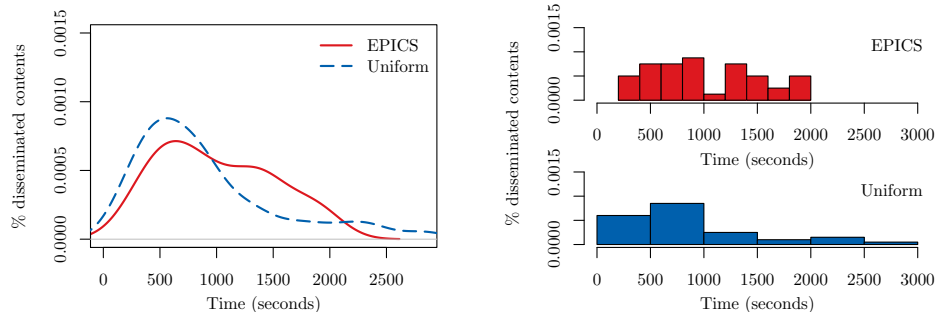
We wonder if both freshness and size metrics impact on the dissemination evolution. To this end, we perform the same experiment by fixing for all the content the same creation time and varying content size (Figure 5(b)) and also by varying the creation time and fixing the content size for all the contents (Figure 5(c)). Again, the same behavior is observed. Although the uniform random selection performs better than EPICS in the beginning, EPICS achieves faster dissemination of all contents at the end.

Dissemination delay distribution. We now want to figure out if the faster dissemination evolution obtained with EPICS is induced by more homogenous content dissemination delays.

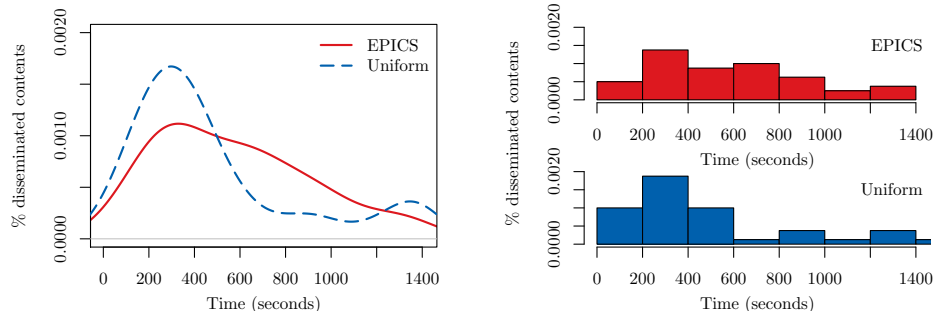
We investigate the distribution of content dissemination delays when either content size or creation time is fixed (Figure 6(a) and 6(b), respectively) and when both size and creation time vary (Figure 6(c)). In Figure 6(c), the distribution with the uniform strategy is skewed. Most of contents have a dissemination delay in the range $[0, 600]$ seconds when remaining ones get very large delays. Similar observation can be made when one metric is fixed. Here again, most of contents have a dissemination delay in the range $[0, 500]$ seconds when contents have fixed size (Figure 6(a)) and in range $[0, 1500]$ seconds when contents are created at the same time (Figure 6(b)). However, as for the first case, the latest contents spend much more time to be fully disseminated. Conversely, EPICS obtains fairer dissemination delays by mitigating the skewness and reducing outliers.



(a) Contents have the same size (140kB).

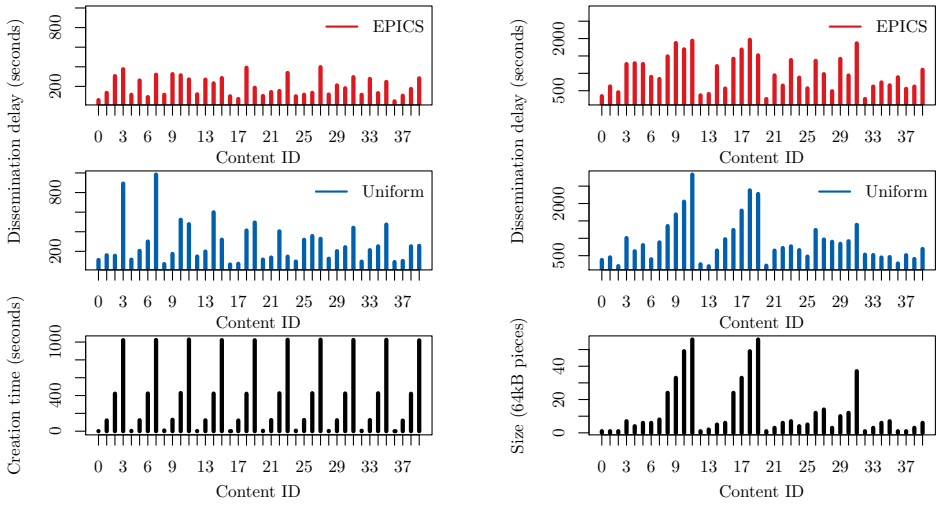


(b) Contents are created simultaneously at time 0 after the warmup period.



(c) Contents have variable size and variable creation time.

Figure 6: Content dissemination delays. EPICS obtains fairer dissemination delays by mitigating the skewness and reducing outliers.



(a) Contents have the same size (140kB).

(b) Contents are created simultaneously at time 0 after the warmup period.

Figure 7: Dissemination delays according to content identifier. By affecting higher weights to larger and fresher contents, EPICS alleviates dissemination delay disparities.

Impact of the creation time and content size on content dissemination delays. Finally, we go into the details of content dissemination delays to find out if the fairer dissemination delays are due to the policy applied in EPICS. Figure 7 shows content dissemination delays according to the identifier of the contents. In Figure 7(a), we vary content creation times and observe dissemination delays accordingly. Each bar in the plots corresponds to a content. The figure at the bottom shows the creation time of each content while the 2 figures above show the dissemination delays of the same content. When compared to the uniform strategy, dissemination delays with EPICS are less correlated to the creation time. Because fresher contents get higher probability to be selected with EPICS, they are more quickly disseminated. In the same manner, when we vary content sizes (Figure 7(b)), uniform strategy dissemination delays show a clear correlation to the content size, while EPICS exhibits more homogeneous delays. By affecting higher weights to larger and fresher contents, EPICS alleviates disparities in the dissemination delays.

3.6 Summary and outlook

Opportunistic content sharing among mobile users is expected to be a widespread application in a near future, as collocated people are likely to share mutual interests. We proposed EPICS, a generic and extensible distributed strategy based on the grey relational analysis for inter-content piece selection when two nodes observe a contact opportunity. We evaluated the performance of EPICS in our testbed composed by 10 Android smartphones and showed that, when compared to a uniform strategy, EPICS ensures fairer dissemination delays for all the contents regardless of their creation times and sizes. As a future research direction, we intend to improve the efficiency of a transfer during a single contact.

4 MOTO terminal-to-terminal content dissemination: EPICS with fine control of burst size

The challenge when handling multiple content files composed of several pieces each is to limit the signaling overhead. Indeed, when two nodes meet, they must first handshake to determine which pieces one can obtain from the other – the larger the number of pieces, the higher the overhead, especially when handshaking after each piece exchange. Existing algorithms in the literature focus on the prioritization of the missing pieces, but do not consider *how many* of them must be sent before handshaking again. It is intuitive to think that it is worth sending as many pieces as possible at once before the connection breaks. Although this holds true in theory and under some assumptions, the story is different in practice. We show that sending a burst of pieces is indeed a good solution *only if the size of the burst is finely tuned*. In a nutshell, the main reason for such a statement is that, if the burst is too long, a node may, meanwhile, retrieve pieces from other nodes. This is due to the fact that, in practice, it is difficult to dequeue packets once they are sent to the output queue at the MAC interface. The consequence is the waste of communication resources with redundant data.

We find out that the node degree is a good metric to determine an appropriate size of the bursts. To this end, we adopt a fully experimental approach. We propose a protocol that dynamically adapts the burst size by referring only to a local measure of node degree. This metric turns out to be a very good indicator (during steady state) of the possibility for a node to obtain a given piece of a content from multiple neighboring nodes. To the best of our knowledge, no previous strategies tackled this specific problem. In order to build up our solution, we adopt a baseline strategy that determines the right pieces to transmit – but that uses *fixed burst sizes*.

We conduct experiments using real devices embedded in an uncontrolled environment. To support performance data analysis, we also rely on wireless traffic traces captured by passive monitors during experiments. We also develop a simulator based on a pedestrian mobility model, to have the possibility of studying the impact of our solution in synthetic traces beyond real mobility traces.

We focus our investigation on the convenience of exchanging bursts in terms of dissemination latency. We clearly observe that sending a large burst of pieces can help or worsen the dissemination, depending on the size of the burst. Thanks to its adaptive behavior, our solution allows much faster dissemination than solutions using fixed burst sizes.

In summary, our contributions are the following:

- We experimentally evaluate the interest of sending a burst of pieces after each handshake. We show that there is a correlation between burst size and density of collocated nodes.
- We propose a protocol that relies on node degree to determine, on the fly, the burst size. Our proposal relies on local information only and does not generate any undesirable signaling overhead.
- We evaluate our solution using real experiments as well as simulations relying on both synthetic and measured mobility datasets. Our analyses confirm the necessity to fine-tune the burst size at each encounter.

4.1 Rationale and problem statement

The motivation behind this work appeared with the implementation of EPICS. As a reminder, EPICS relies on a local measure of "prevalence" to decide which piece to transmit first. Upon a contact, a node decides to first send pieces that, from its local point of view, are less prevalent in the network. The prevalence is computed based on a counter that indicates how many times the node has detected a copy of a given piece at its neighbors. It involves a handshaking phase where a node sends to its neighbor the list of pieces it misses.

Because nodes are likely to experience contacts with multiple neighbors at the same time, the baseline functioning of EPICS performs handshaking after the transmission of each piece. During the experiments, we noticed that performance changed significantly depending on the network conditions. In particular, we figured out that handshaking after each transmission was the best configuration in dense networks but did not perform well in sparse setups. We decided to investigate this issue further and the results are reported in the following.

In the rest of this section, we will also call EPICS as "fixed strategy" and our proposal "adaptive strategy". We refer to the "baseline" approach as the case of a burst of size = 1.

4.2 Experimental assessment of the impact of burst size

Our work adopts a fully experimental approach – as we will discuss later, the performance degradation we observed in our tests are due to practical aspects that are not captured by existing models. Before investigating the impact of the burst size on the behavior of the system, let us first describe the testbed we considered in our experiments.

4.2.1 Testbed

We conduct experiments in a real, uncontrolled environment of smartphones running Android and configured in IEEE 802.11 ad hoc mode [6]. The system can be easily scaled up since it is also available for Android-x86 running as a virtual machine [7]. In our experiments, we use eight Samsung Galaxy SII smartphones. We setup the dissemination protocol to chop content in pieces of 25 Kbytes. During a calibration phase, we found out that 25 Kbytes is an appropriate piece size, in the range [5-64] Kbytes, in order to obtain fast dissemination [8].

Given that we use real devices in a real environment, we decided to run a passive wireless monitoring system to capture events and traffic generated by surrounding devices, helping us to understand the experiments in detail. We use WiPal both as wireless capturing and trace merging software [9, 10].

4.2.2 Impact of burst size

First, we investigate is if it is worth sending a burst of *least prevalent* pieces at each contact. We set up our testbed with one source node which has five contents of 3 Mbytes each and seven nodes, each requesting all the contents. As soon as a node gets a piece, it also acts as a source in a P2P fashion. The burst size is modulated in the set of {2, 3, 5, 10} pieces. The basic design of EPICS uses bursts of size = 1.

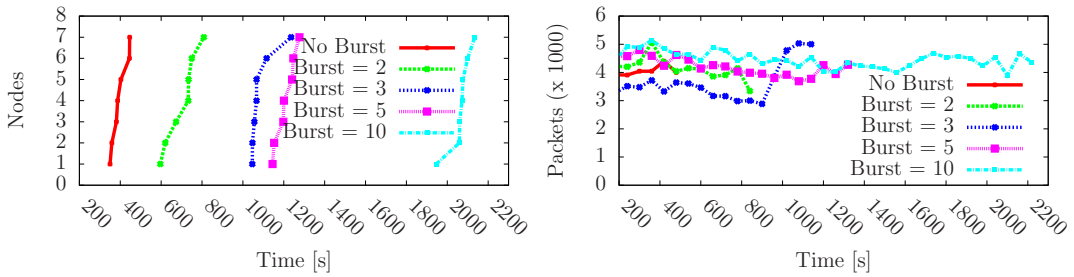
Performance, in terms of dissemination latency, becomes worse and worse as the burst size increases. Fig. 9(a) shows, for each burst size, the elapsed time to achieve a complete dissemination of the five contents to one node, two nodes, until all the seven nodes which requested

Figure 8: At the bottom, off-the-shelf Android devices running our protocol. At the top, two passive monitors with three antennas capturing environmental wireless traffic.

the contents. In many cases, with a burst size = n the dissemination on the seven nodes is faster than the same dissemination on only one node with a burst size = $n + x$. These results are counter-intuitive, as they are completely independent from the external wireless traffic listened by the monitors (Fig. 9(b)). For example, when burst size = 3, we record less external traffic than in the case we do not use any burst; it takes, however, exactly three times more to disseminate the contents.

Fig. 10 shows the piece transmission efficiency by tuning the burst size. The efficiency is calculated as the percentage of the total amount of pieces required by all the nodes over total number of pieces placed in the transmission queue by all the nodes. As an example, let us consider that 7 nodes require 5 contents of 126 pieces each. Thus, in order to complete the dissemination, $7 \times 5 \times 126 = 4410$ right pieces must be received among all nodes. If 8820 pieces have been sent instead, the efficiency is equal to 50%. With no burst, the efficiency is around 70%. The remaining 30% efficiency decrease is due to pieces lost during transmission and then retransmitted, and by duplicate pieces, that are pieces well received but useless, since already received from other nodes before. In the analysis of wireless traces, we have not found many retransmitted pieces, so the efficiency is mostly affected by duplicate pieces. With a burst size = 2, the efficiency drastically decreases to almost 20%, and continues to decrease with larger burst sizes. Moreover, pieces experience longer queue delays, enlarging the burst.

Let us assume a burst size = 10 as shown in Fig. 11. At each contact with another node, at most ten pieces are placed in the transmission queue. The queue grows ten times faster than the basic solution without burst. These ten pieces are chosen based on a local and contemporary view. The transmission queue is FIFO, without preemption. For each piece included in a data-link frame, a node must gain access to the wireless medium waiting to be idle or reserving a slot



(a) Dissemination latency of five contents on seven nodes modulating the burst.

(b) External traffic observed during the experiments.

Figure 9: Investigation of dissemination time in function of the burst size. To be sure the results are not influenced by external surrounding wireless traffic, we also monitor the environment with a passive monitoring system.

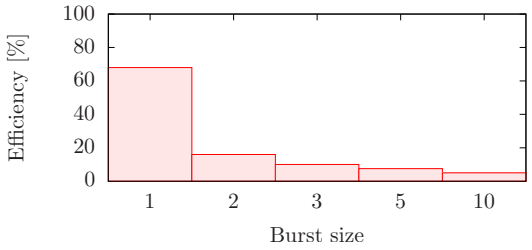


Figure 10: Piece transmission efficiency tuning the burst size.

with the RTS–CTS mechanism. Thus, when pieces in the tail of the queue (e.g., pieces to node 3 in the figure) eventually reach the head of the queue, they are likely to be obsolete, wasting transmission slots (i.e., other neighbors may have already sent that piece to the node).

We deeper investigate this point varying the quantity of nodes involved in the content exchange. We make an experiment sharing one content of 3 Mbytes with only one source and one node requiring the content (node degree = 1). Then, we make the same with one source and two other nodes (node degree = 2), until there are eight nodes in total (node degree = 7). For each experiment, we tune the burst size from one to ten and we repeat it three times in different hours of days. Fig. 12 shows the average dissemination latency in each case. We note that dissemination is faster using a large burst when only few nodes are employed. On the other hand, with more nodes in contact, dissemination is faster by reducing or disabling the burst. As we can see, with node degree = 1, the minimum latency is achieved with the maximum burst size = 10. With a node degree = 2, the minimum values of dissemination delay are expected with a burst size = 4. With larger degrees, it is worth using smaller burst sizes, until the node degree = 5, where the minimum latency is achieved without bursts.

4.3 Fine Tuning Burst Sizes for Fast Opportunistic Dissemination of Large Files

In Fig. 13 we connect burst size values, per source node's degree, achieving the minimum dissemination delay in Fig. 12. The gray area includes values of burst for a diffusion time at most 30 seconds longer than the minimum. Note that this area becomes narrower and narrower as

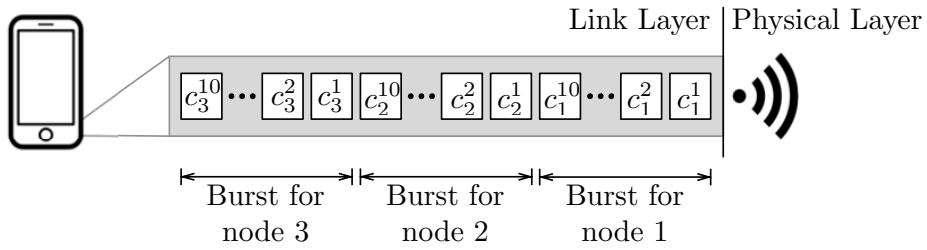


Figure 11: View of a node's transmission queue using a burst size = 10. Note that pieces experience a long queue delay becoming outdated once transmitted.

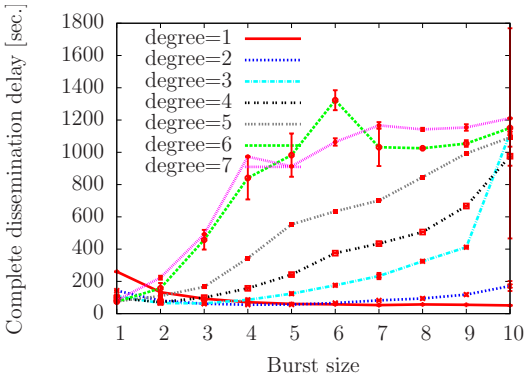


Figure 12: Complete dissemination delay average and standard deviation, tuning the burst size and the nodes degree.

the number of nodes in contact grows. In this plot, the fixed burst size solution moves on the bottom, meaning that it can be improved, up to a node degree of four. From a node degree of five up, it is worth sharing only one piece per contact handshake.

As a consequence, we advocate for the inclusion of some flexibility in the choice of the burst size. Based on the observations reported in the previous sections, we propose a degree-based adaptive dissemination solution that modulates the burst size according to the number of neighbors, always following the minimum diffusion time line of Fig. 13 and shown in the following abacus:

Abacus

node degree	1	2	3	4	≥5
burst size	10	4	3	2	1

We compare the adaptive solution against the baseline fixed version (i.e., with a burst size = 1) and against an extreme case with fixed bursts of ten pieces. We start this experiment with only two nodes: one source that has 10 contents of 3 Mbytes to share and another node. Then, every three minutes we add a new node requiring all of the contents, up to seven nodes. We gather relative completion times for each content from every node and we present the cumulative distribution function in Fig. 14. Even if both approaches take the same time to diffuse all the contents to all the nodes since from 5 nodes up in the network they have the same mode of

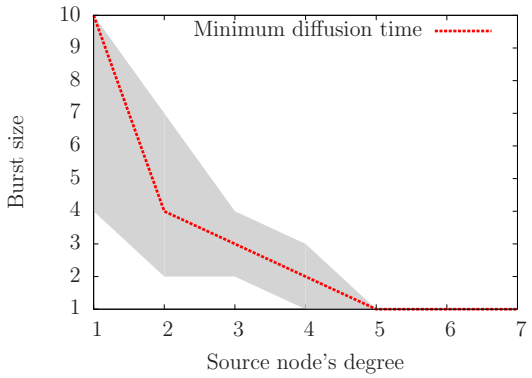


Figure 13: Operation area for an adaptive solution. Line connects burst size values for minimum diffusion time. Gray area includes dissemination delay values 30 seconds longer than the minimum.

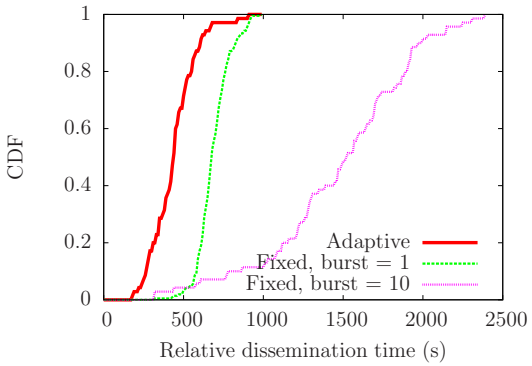


Figure 14: Adaptive vs. fixed with burst sizes of sizes 1 and 10. The y-axis shows the CDF of relative dissemination time per content and per node.

operation, they present a considerable difference until the 97^{th} percentile. It means that the dynamic adaptation not only facilitates the diffusion when there are only a few nodes, but, since the content is almost fully received in many nodes, these nodes can better support the dissemination even when we insert more nodes in the network. On the other hand, when using fixed bursts with the maximum size, the dissemination is very fast for the first few contents (i.e., when there are only few nodes), and then it slows down taking five times longer.

We also compare the adaptive versus the fixed solutions with static node degrees (Fig. 15). In each case, at the beginning there is only one source node with a content of 3 Mbytes. A node degree of one reveals the largest difference (the adaptive approach is 5 times faster) between the two approaches. The difference becomes narrower as the node degree increases, but the adaptive solution still remains three and two times faster.

4.4 Using adaptive bursts in practice: Real and synthetic mobility traces

We showed in Section 4.3 how tuning the burst size can either improve or worsen the dissemination performance depending on the number of nodes in contact. To check how important

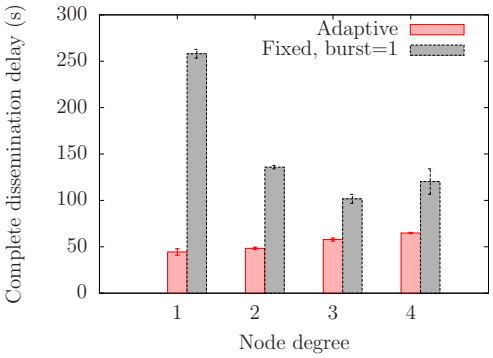


Figure 15: Complete dissemination delay with respect to different node degrees.

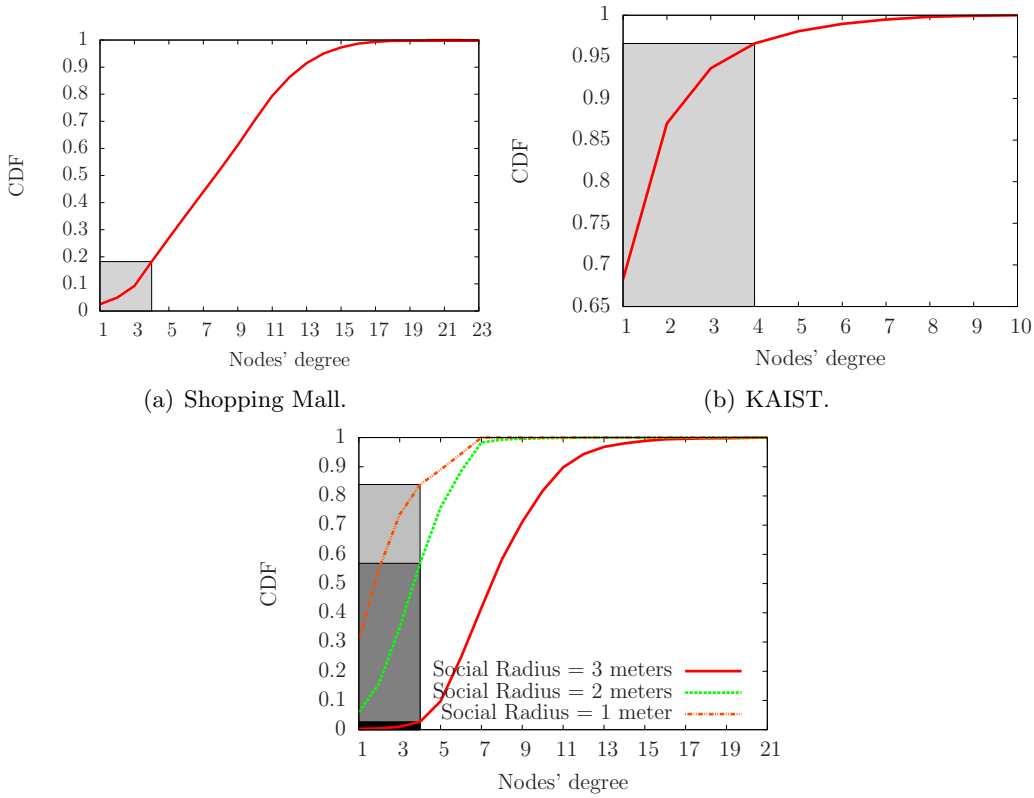
this profit margin is, we analyze in the following, some real and synthetic mobility traces. In particular, we examine the cumulative distribution function of nodes degree at every beaconing instant. We exclude from the distribution isolated nodes, as they cannot exchange contents with anyone.

We consider the following mobility traces:

- **Shopping Mall [11].** This is a six days dataset of real-world Bluetooth contact data collected from a mall in Nottingham (UK). 25 devices captured Bluetooth contacts.
- **KAIST [12].** Another real-world dataset, consisting of 92 daily GPS track logs collected from the KAIST university campus in South Korea. Traces have been overlapped in time to produce one single trace. We assume a contact range of 10 meters, compatible with the Bluetooth trace.
- **SIMPS (synthetic) [13].** We also consider a mobility model of human crowds with pedestrian motion called SIMPS and we implemented a simulator based on this model. We simulated a relatively dense toroidal space of 100×200 meters with 100 people moving around for one hour. This model is based, among other parameters, on a "social radius". Nodes take decisions about their movements according to the nodes they detect in that radius. In crowded environments, the social radius tends to shrink. Since we simulated a crowd model, we used a social radius of one, two, and three meters, varying the contact range accordingly.

We chose these traces because they are different in many aspects: nature (real and simulated), environment (mall, university campus, simulated toroidal plane), log collection (Bluetooth device, GPS, 2D position), plane size (medium, huge, small), and density (medium, low, high). Moreover, being pedestrian mobility traces, we believe that contact duration is enough to exchange the maximum burst of 10 pieces.

Fig. 16 shows the distribution of node degrees for the traces we consider. The probability to find a node with a degree, at most, equal to four, considerably varies from trace to trace. The Shopping Mall in Nottingham has an indoor surface area of 10,880 square meters. We can image it very crowded, especially during rush hours. Nonetheless, not everyone has a Bluetooth enabled device. Thus, we can detect nodes having a degree, at most, equal to 4, with about 20% probability.



(a) Shopping Mall.

(b) KAIST.

(c) SIMPS (synthetic).

Figure 16: Cumulative distribution function of the number of nodes that every node perceives in its neighborhood.

The KAIST campus has a 1,432,882 square-meter area. This big scenario makes it possible to exhibit a very high probability (more than 95%) of nodes with a degree = 4 at most. For the sake of fairness, being a GPS trace, indoor places that should be the most crowded are not taken into account. In this case, using a large burst will largely improve the outdoor opportunistic exchange of content.

We have also simulated a very dense scenario using the SIMPS mobility model (with 100 people in a 100×200 m^2 plane). The improving margin considerably changes for slightly changes of social radius. In the case of three meters, there is only a 0.03% of improving margin. For a social radius of two and three meters, we get nodes with a degree of at most 4 with a probability of 55% and 85%, respectively.

As we can see, there is room for improvement in all cases. Furthermore, the gains can be significant. This means that the use of an adaptive burst size selection mechanism is recommended, as *the operation point observed in practical scenarios frequently falls in the* $[1-4]$*-degree range.*

4.5 Summary and outlook

Starting from an opportunistic multi-content dissemination protocol based on content chopping and piece selection, we investigate the convenience to transmit a burst of pieces at each opportunistic contact. To be sure not neglecting aspects related to wireless communication mecha-

nisms, we simultaneously deploy a testbed with real off-the-shelf devices and passive monitors to capture the generated traffic. Results show the need for dynamic burst dimensioning based on the node degree.

We found out that, the lower the neighborhood cardinality, the larger the burst size should be. In addition, when the degree is above 4, it is worth sending only one piece per handshake. Indeed, results show that using large bursts in crowded environments leads to a significant amount of redundant transmissions. Pieces experiencing long queuing delay are most likelihood to be sent earlier by other nodes, leading to duplicate exchange and loss of transmission slots. After comparing our adaptive solution against a basic strategy that only sends one piece per contact and against a strategy that always uses a static large burst, we investigated its impact analyzing some real and simulated mobility traces. This analysis revealed that our proposal has enough potential to yield significant gains in multiple scenarios, and in particular in sparse environments.

Future research directions include the extension of our proposal by taking into account other parameters such as the piece size, the number content files, and lower level transmission rates.

5 Choosing the right seeders

While opportunistic networks offer additional capacity that can be leveraged to reduce congestion on the cellular network, timely delivery of content is an issue, due to the variability of human mobility and the resulting stochastic nature of forwarding events. When offloaded content must be delivered within given deadlines, we need offloading solutions that both meet these deadlines, and reduce as much as possible the traffic carried by the cellular network.

We assume a typical model for this type of offloading [14], whereby a content dissemination controller decides dynamically over time to which nodes content must be sent through the cellular network, based on the current status of the dissemination process. This is tracked through ACKs sent by nodes (over the cellular network) upon receiving content (over the opportunistic network). Injections of content to specific nodes through the cellular network helps to boost the dissemination process over the opportunistic network, eventually helping to meet delivery deadlines with minimal traffic over the cellular network.

We design a novel solution to control the above-described dissemination process.[2] Specifically, the controller has to cope with two problems: (i) identifying how many nodes should be delegated to disseminate a content and (ii) which of them are more useful to speed up the dissemination process. Due to high dynamicity of the opportunistic dissemination process, it would be desirable that the central dissemination controller would be able to cope with this problem autonomously, in order to reduce as much as possible the human intervention.

Therefore, we solve the first problem through a Reinforcement Learning (RL) approach. Precisely, we modelled the offloading problem as a RL problem, in a very general way such that we can apply and try many different RL learning algorithms. Specifically, in our context a learning agent (the controller) interacts with an unknown environment (the opportunistic network). For each content, at fixed time steps, the agent observes the state of the *system* (i.e., the current diffusion of the content), takes an *action* (i.e., decides whether to inject new content replicas) and receives a *reward*, i.e., a feedback to evaluate the action taken. Its goal is to learn

[2] L. Valerio, R. Bruno, and A. Passarella, "Adaptive Data Offloading in Opportunistic Networks through an Actor-Critic Learning Method", *ACM Workshop on Challenged Networks*, Maui, Hawaii, USA, September 2014.

a *policy* (i.e., a mapping from states to actions) that maximises the long term reward.

Among the many existing RL techniques, we used and evaluated two well known and very robust solutions: the Actor Critic and the Q-Learning algorithms [15, 16]. For nodes selection we adopted a heuristic mechanism that permits to identify, online, what nodes are more useful to spread contents. Precisely, exploiting some additional information contained in the ACK messages, the controller tracks what nodes contributed the most to the content diffusion forming a priority list to be used for future nodes selection. Through a thorough set of simulative experiments we will show that not all the RL algorithm produce the same good performance. We anticipate that Actor Critic proves to be more suitable than Q-Learning in this specific problem. Moreover, it will be clear that depending of the capabilities of the learning algorithm to cope with such difficult problem, combined with the seeds selection mechanism, could make the offloading solution very efficient, many different mobility patterns and with many different constraints on content delivery deadlines. In addition, our results will show that a solution based on RL does not require pre-existing knowledge about the nodes behavior (e.g., degree on the contact graph), but is able to automatically learn the most appropriate control strategy for the offloading task.

5.1 Positioning

In this last years, much effort has been devoted in finding good solutions to cope with the problem of data offloading in cellular networks. In this section we comment the main results regarding data offloading solutions based on opportunistic networks for delay tolerant content. A more complete and exhaustive survey on state of art of data offloading techniques in cellular networks can be found in [17].

To the best of our knowledge, Han et al. [18] (then subsequently extended in [19]) were the first to exploit opportunistic communication to alleviate data traffic in the cellular network. In their work they propose and evaluate three algorithms, i.e. *Greedy, Heuristic* and *Random* that exploit information about the contact graph of mobile nodes, in order to select the initial set of k mobile nodes in charge to trigger the opportunistic dissemination of the requested content. Precisely, in the Greedy and Heuristic algorithms, by exploiting information about contacts, they compute the dissemination power of each user in the opportunistic network and the first k most powerful ones are selected. Finally, with the Random approach, k users are randomly selected.

Li et al. propose to treat the selection of seed as a maximisation problem in which the utility function to be optimized is subject to multiple linear constraints such as the mobility of users, traffic heterogeneity and the storage available on mobile nodes [20]. These two approaches share a common point: the network controller has to collect information about nodes' contact rates in order to come up with the best set of seed nodes.

A similar (in principle) solution proposed by [21] suggests to exploit social aspects of user mobility to select what they call VIP devices that are in charge to spread contents in the opportunistic network. VIP devices are the most socially important nodes in the network, i.e. they are able to eventually contact the rest of the network. They assume that, among subscribers in the network, some of them are more socially important than others and through their movements they are able to eventually contact the rest of the network. Precisely, every time a big amount of data must be transmitted to interested users, VIP devices act as bridges between the network controller and the opportunistic network. Authors identify two kind of

VIPs: *global* and *local*. The former are users globally important in the network while the latter are important within their social communities. The social importance of users is estimated through well known metrics like betweenness, degree, closeness and Page Rank.

Following similar principles, Chuang and Lin take into account the social relationship between users of an opportunistic network considering also information about their social communities in order to select the initial set of users through which bootstrap the spreading of content in the opportunistic network [22]. Precisely, authors try to minimise both the number of initial seed users and the latency of the opportunistic dissemination. To this end they propose community-based opportunistic dissemination, which automatically selects a sufficient number of initial sources to propagate the object across disjointed communities in parallel. In this schema, mobile end-users are required to upload periodic information on the most frequent contacts, in order to let the centralized algorithm to choose the best subset of seed users.

These approaches are strongly based on the exploitation of social information collected from the network to select an initial set of users to be used to trigger the opportunistic dissemination process. However, after initial injection of contents, no other actions are taken.

A solution based on mobility prediction (TOMP) is proposed by Baier et al. Authors try to predict users' mobility in order to estimate the future connectivity opportunities between mobile nodes. To this end, TOMP (Traffic Offloading with Movement Predictions) exploits information about actual positioning and speed of mobile devices rather than connectivity patterns. The framework selects as seed users the nodes that have the best future connectivity likelihood with other nodes based on movement prediction. TOMP proposes three coverage metrics to predict the future movements of nodes: static coverage, free- space coverage, and graph-based coverage.

A quite different approach is proposed in [14]. Authors propose Push&Track (PT), an offloading solution based on predefined performance targets used by the network controller to compute the number of content replicas to inject. Precisely, they designed a control loop based on ACKs notifying the controller of the delivery of a content to a user. Precisely, the controller, at fixed time steps, operates content re-injections in order to force the opportunistic dissemination trend to follow a target dissemination evolution (i.e. a predefined target percentage of dissemination at given points in time). Beyond the number of seed nodes to be selected during time, the authors investigate methods for nodes selection like geographic position or the connectivity degree. The evolution of PT is represented by Droid (Derivative Re-injection to Offload Data) [23], which, periodically, adaptively estimates the actual dissemination trend, evaluates the evolution of dissemination level for future time steps and, if needed, injects new content replicas in order to reach the total diffusion within a time limit.

In Droid the authors keep unchanged the ACKs mechanism of PT but leave the predefined dissemination targets in favor to an adaptive content offloading mechanism. Precisely, at fixed time steps the controller re-calculates (exploiting the ACKs received) the ideal trend to reach the full coverage within a maximum time limit. Differently from PT, the controller using a derivative approach, estimates the actual dissemination trend, evaluates the evolution of dissemination level for future time steps and checks if with the current trend of content dissemination will reach the total diffusion within the time limit. Then, it exploits this information to compute the number of content replicas to be re-transmitted in the network. Due to its adaptive and incremental injection strategy, Droid represents a natural benchmark for our solution, hence, we compare our results with Droid.

Almost all of the offloading solutions in the literature consider significant information about the social relationship between nodes in order to select a proper subset to which delegate part

of the content distribution. Moreover, except for PT and Droid, just an initial set of nodes is selected with the consequence of not having any control on the evolution of the dissemination process. However, in solutions like Droid, decisions of re-injection are taken using current information only.

Instead, we argue that the controller can obtain better results if it learns by experience what is the most appropriate sequence of re-injection decisions to maximise the coverage while minimizing the cellular network load. Precisely, we exploit a learning mechanism able to abstract from the single scenario and learn from experience what is the most appropriate injection policy. In this way, our solution reaches, autonomously and independently from the mobility scenario, very high levels of offloading.

5.2 Problem statement and system assumptions

We consider a system composed by N mobile nodes equipped with two network interfaces: Wifi and 3G /LTE. The former is devoted to the opportunistic communication among devices and the latter is used to communicate with the cellular controller. We assume that nodes are interested in receiving a content. Here we consider contents with a limited validity, thus they must be delivered to users within a deadline.

We assume a central dissemination controller that decides whether to reach interested users directly through the cellular network or indirectly through opportunistic networking dissemination. The controller has to guarantee deadlines while minimising traffic over the cellular network.

To this end, the controller delegates few mobile nodes (from now on called *seeds*) to opportunistically disseminate the content to all the other peers in the network. Without loss of generality we may assume that the opportunistic dissemination between nodes is done by means of an epidemic algorithm [24]. Once a content is delivered to a node, the node acknowledges the controller sending a message containing its ID and two additional information: the ID of the seed selected by the central controller who started the diffusion of that content replica, and the ID of the last forwarder. Since the ACK message is quite lightweight (compared to the content size), the impact of this feedback mechanism can be considered negligible from the network load point of view.

After the initial content transmission to a limited subset of mobile nodes (seeding), at fixed time steps the controller evaluates the level of dissemination together with the remaining life time of the content and decides whether or not to operate a re-injection of the same content in the opportunistic network. Thus, every time step the controller calculates how many mobile nodes should be seeded with the content. If the opportunistic dissemination can not reach all interested nodes within the time limit, the controller enters in a "panic zone" and pushes the content to all the remaining nodes through the cellular network (similarly to [14]).

5.3 Reinforcement learning background

In this section we describe the general principles of the Reinforcement Learning (RL) algorithm we used in our approach.

A problem of RL can be modeled as a Markov Decision Process defined by the tuple $\langle \mathcal{X}, \mathcal{A}, \phi, \rho \rangle$ where $\mathcal{X}$ is the set of states, $\mathcal{A}$ is the set of actions, $\rho : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function and $\phi : \mathcal{X} \times \mathcal{A} \rightarrow \mathcal{X}$ is the state transition function. At each time step, the reward

function specifies the instantaneous reward obtained by taking a given action in a given state while ϕ represents the state transition triggered by the action. Precisely, the ϕ function returns the state x_t that the system reaches from state x_{t-1} after applying action a_{t-1}. At each time step the actions are taken according to another function called *policy* $\pi : \mathcal{X} \rightarrow \mathcal{A}$. Note that in our problem the function ϕ is unknown to the controller because it cannot know in advance the real effect of a content injection on the dissemination process in the opportunistic network. Thus, the controller just observes a series of state transitions and rewards. Its goal is to find a policy such that the sum of future rewards is maximized. The quality of a policy is quantified by the value function $V^{\pi} : \mathcal{X} \rightarrow \mathbb{R}$ defined as the expected discounted cumulative reward starting in a state x and then following the policy π:

$$V^{\pi}(x) = \sum_{t=0}^{\infty} \gamma^j r_t, \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor. The goal of the controller is to learn a policy such that for each state x the sum of future rewards is maximized:

$$V^{\pi *}(x) = \max_{\pi} E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right). \tag{2}$$

Intuitively, the value functions evaluates a policy in terms of the reward it is expected to provide, starting from a given state x. Future rewards are weighted with a discount factor γ to take into account their uncertainty (i.e., higher discounts are used for rewards expected further away in time). As will be clear in the following, practically the value function must be estimated, as future rewards are not known in advance, as we will show later on.

We consider two well known and solid model-free approaches of Reinforcement learning: Actor-Critic Learning and Q-Learning. In the following we introduce both of them, providing details about how they are applied in the context of cellular data offloading.

5.4 Actor-Critic Learning

The actor-critic methods are policy-gradient methods based on the simultaneous on-line estimation of the parameters of two structures: the *Actor* and the *Critic*. The Actor corresponds to an *action-selection policy* that maps states to actions in a probabilistic manner, i.e. depending on the current state, it tries to select the best action to take according to the policy – there is a different best action for each state. The Critic is the *state-value function* that maps states to expected cumulative future rewards. Basically, the Critic actually is a function predictor that during time learns the value of a state in terms of the expected cumulative rewards for that state. Therefore, the critic solves a prediction problem, while the actor solves a control problem. Although separate problems, they are solved simultaneously in order to find an optimal policy. In Actor-Critic methods the Critic evaluates the actions taken by the Actor and drives the learning process in order to maximise the cumulative rewards. In the following we describe both the Critic and the Actor separately.

5.4.1 Critic

In our proposed offloading scheme the Critic is designed by exploiting a *temporal-difference* (TD) learning method. TD learning is a well known incremental method introduced by Sutton [15] to

estimate the expected return for a given state. Specifically, when the transition from the state x_{t-1} to the state x_t and the reward r_t are obtained, according to the TD theory, the estimation of the value of the state x_{t-1} is updated in the following way:

$$\hat{V}^{\pi}_{new}(x_{t-1}) = \hat{V}^{\pi}_{old}(x_{t-1}) + \alpha \left[r_t + \gamma \hat{V}^{\pi}(x_t) - \hat{V}^{\pi}_{old}(x_{t-1}) \right] \quad (3)$$

where $\alpha, \gamma \in [0, 1]$ are discount parameters. The rational of this update equation is that the value of each state is given, on the one side, by the reward derived from taking a given action in that state and, on the other side, by the difference among the values of the states x_{t-1} and x_t. The idea behind this method is to evaluate each state in term of its "potential", i.e. the higher the reward obtained and the higher the difference among states' values the more valuable the state x_{t-1} becomes. At the beginning, state values are initialized to 0. The role of the Critic is to evaluate the utility of the selected action a_{t-1} taken in the state x_{t-1}. This utility value is calculated as follows:

$$\delta = r_t + \gamma \hat{V}^{\pi}(x_t) - \hat{V}^{\pi}_{old}(x_{t-1}) \quad (4)$$

If $\delta >> 0$ the action just taken was good because it brought the system in a much better state. As it will be clear from Section 5.4.2, this is provided as a feedback to the Actor on the effectiveness of the action taken at time $t-1$ when being in state x_{t-1}.

5.4.2 Actor

Now we describe the policy function through which actions are selected. As we want to explore and find what are the more profitable actions to take for each state, we choose a probabilistic action selection method. Precisely, we used the Gibbs Softmax method [15], a well known approach in RL for action selection. According to it, for each state x_t, the probability of selecting an action a_t is given by:

$$\pi_t(x_t, a_t) = \frac{e^{p(x_t, a_t)}}{\sum_{b_t \in \mathcal{A}} e^{p(x_t, b_t)}} \quad (5)$$

Thus, in order to select an action the Actor draws a sample from the above distribution. The probability value of each action is affected by the exponent of the exponential function, denoted by the function $p(x_t, a_t)$. This function is typically called *preference function* and represents the weight of each action, i.e. an action connected with a higher preference value than the other ones has more chances to be selected. Therefore, learning a policy consists in strengthening or weakening the value of $p(x_t, a_t)$. In the Actor Critic framework, this is done in the following way:

$$p_{new}(x_t, a_t) = p_{old}(x_t, a_t) + \beta\delta \quad (6)$$

where $0 < \beta \leq 1$. At the beginning, values of p for all actions are initialized to 0. As we can notice, the preference value is updated according to the evaluation made by the Critic, or in other words, the Critic drives the future actions of the Actor.

Summarizing, the entire algorithmic procedure for the Actor Critic method is as follows. At time t the system is in the state x_t. The Actor draws and executes an action a_t according to (5). When the new state x_{t+1} and the reward value r_{t+1} are observed, the Critic computes δ as in (4) and updates the estimation of the value function $\hat{V}^{\pi}(x_t)$ according to (3). Finally, the policy for state x_t is updated as in (6). This procedure is repeated for every time step.

5.5 Q-Learning

Q-Learning (QL) has been firstly introduced by Watkins [16]. In principle, Q-Learning is similar to Sutton's TD learning: an agent tries an action at a particular state, and evaluates its consequences in terms of the immediate reward it receives and its estimate of the value of the state to which it is taken. Differently from the Actor Critic method presented before, Q-Learning does not evaluate the value connected to a state, instead, it evaluates the pair *state-action*. In fact Q-Learning does not estimate the function $V^{\pi}(x)$ like in the Actor Critic but the function $Q^{\pi}(x, a)$. Formally, after every state transition, the value for the state action pair is updated as follows:

$$\hat{Q}^{\pi}_{new}(x_{t-1}, a_{t-1}) = \hat{Q}^{\pi}_{old}(x_{t-1}, a_{t-1}) + \alpha \left[r_t + \gamma \max_a \hat{Q}^{\pi}(x_t, a) - \hat{Q}^{\pi}_{old}(x_{t-1}, a_{t-1}) \right] \tag{7}$$

where $0 < \alpha, \gamma < 1$ are discount parameters. Specifically, the function $Q^{\pi}(x, a)$ evaluate the expected return if the agent starts at state x, executes action a and follows the policy π. It is worth noting that Q-Learning, under very general assumptions, asymptotically converges in probability to the optimal action-value function $Q*$ independently from the policy π. However, the algorithm converges after number of time steps that goes to infinity, meaning that the rate of convergence is very slow. Moreover, in order to converge each action must be tried many times to gain a reliable estimate of its expected reward.

In this work, Q-Learning algorithm is used in combination with two different action selection policies: ε-greedy and *Softmax*.

The ε-greedy is one of the simplest action selection policy. According to this policy, when the system is in a state x, the controller selects with probability $(1 - \varepsilon)$ the action with the maximum accumulated reward for that state and with probability ε one of the other actions at random (uniformly), independently of the reward estimates. More formally:

$$\pi(x) = \begin{cases} \text{draw random action} & \text{, with prob. } \varepsilon \\ \max_a Q(x, a) & \text{, with prob. } 1 - \varepsilon \end{cases} \tag{8}$$

The Softmax policy is the same presented for the Actor Critic approach in Section 5.4.2 with the only difference that here the exponent of the exponential function is the Q function. In fact, in the Q-Learning version each action a is weighted according to its value for the state x (and not the value of the state as in the Actor-Critic algorithm). Formally, the probability of the action a for the state x is defined as:

$$\pi(x_t, a_t) = \frac{e^{Q(x_t, a_t)}}{\sum_{\forall a' \in \mathcal{A}} e^{Q(x_t, a'_t)}} \tag{9}$$

Summarizing, the algorithmic procedure for the Q-Learning method is the following. At time t the system is in the state x_t. The action a is drawn according to policy (8) or (9). When the

new state x_{t+1} and the reward value r_{t+1} are observed, the action-values for state x_t are updated according to (7). This procedure is repeated for every time step. As we can notice, compared with AC this algorithmic procedure is very simple to implement and this is one of the reasons for the wide diffusion of Q-Learning, although its performance are not always satisfactory, as we will show in Section 5.7.

5.6 Offloading through Reinforcement Learning

In this section we describe how we exploit the RL framework in the context of data offloading through opportunistic networking. In the following we present a general procedure in which both AC and Q-Learning can be plugged in without any further modification.

In order to apply both RL algorithms in this context we need to define: the system state representation, the actions allowed for AC and QL and the reward function. In our solution the system state at time t is represented by the three dimensional continuous vectors $\boldsymbol{x}_t = \{x_{1_t}, x_{2_t}, x_{3_t}\} \in [0,1]^3$ where x_{1_t} represents the dissemination level for the content, x_{2_t} is the fraction of currently used seeds w.r.t. the total number of interested nodes (i.e. this quantity is less than or equal to x_{1_t} depending on the effectiveness of the opportunistic dissemination) and x_{3_t} denotes the percentage of remaining time for content delivery before the panic zone. In our solution the actions represent the percentage of nodes (still waiting the content) to be used for the next re-injection. In this specific problem we used a discrete set of 11 actions: $A = \{a_1, a_2, a_3, \ldots, a_{11}\} = \{0, 0.01, 0.02, \ldots, 0.1\}$, i.e. we consider re-injections from 0% to 10% of the (remaining) nodes that should get the content, in steps of 1%. Note the chosen set of action is just one possible setting, any set of action are possible, depending on the context. The reward used to evaluate the action taken by the learning algorithm is computed as follows:

$$r_t = \omega(1 - (x_{1_t} - x_{2_t}) + (1 - \omega)(1 - x_{3_t}) \tag{10}$$

where $0 \leq \omega \leq 1$. The first term in (10) measures how much the dissemination process on the opportunistic network has been effective up to time t. In fact $1 - (x_{1_t} - x_{2_t})$ is the fraction of nodes that have not been reached by the opportunistic dissemination process. This component, therefore, penalizes actions that lead to using too many seeds (i.e. to low values of $x_1 - x_2$). The second component considers the relative time we still have until the content deadline. It's role is to give more value to a good action if it is taken closer to the deadline, i.e. when it is needed most. Note that equation (10) is compliant with the definition of the reward function, i.e., it is a function of both the state where we start from at time $t-1$, and the action taken, as the action leads to the values of the state at time t, which are used to compute the reward. Note that the objective of both Actor-Critic and QL algorithms is to maximise rewards, therefore in our system we consider $-r_t$ instead of r_t that, in practice, it is equivalent to minimise equation (10).

During the validity of a requested content c, our system operates multiple injections of c in the opportunistic network to speed up the content diffusion. Now we will describe the entire procedure adopted by the controller to drive the opportunistic dissemination process. For the sake of clarity, we will refer to Algorithm 3.

First, the system selects and transmits the content c to an initial subset of interested nodes in order to trigger the opportunistic dissemination process. In order to select the initial set of seeds the RL algorithm draws an action a according to the selected policy π (line 6). Precisely,

in the AC approach the Actor, i.e. the action selection policy, draws an action according to equation (5), while in QL is used equation (8) or (9).

Once the action is selected, our system calculates the number of new seeds, selects them, and sends them the content (lines 8-9). The seeds selection is done considering the priority of each node. Precisely, each interested node in the system is associated to a preference value. Every time a content is delivered to an interested node, the latter notifies the delivery to the controller together with the ID of the seed associated to the content and the ID of last forwarder. Then, the controller updates the counters associated to those nodes. In this way, during time the controller can estimate the infection power of each node and exploit this information to identify the most infective nodes to be selected as new seeds (lines 26-29). We point out that in this context the term"infection power" refers to the ability of a node of disseminating a piece of content in the opportunistic network. Specifically, the more peers receive a content from a node the more infection power the node has. Second, every fixed time step t the system observes the new state $\boldsymbol{x}_t$ and the reward value r_t according to equation (10). It uses this information to update the estimate of the value function $\hat{V}^{\pi}$ (or $\hat{Q}^{\pi}$ in the case of QL) for the state $\boldsymbol{x}_{t-1}$ (lines 16-18) and the policy parameters associated to that state (line 19), i.e. the preference value for each action. Once the state-value function (or the action-value function for the QL algorithm) and the policy are updated, we use the latter to draw the next action, i.e. the new portion of seeds to which transmit the content (line 20). Finally, new seeds are selected according to the procedure previously described (line 21-22). Third, when the dissemination process enters in the panic zone (line 11) the controller transmits the content to all the remaining interested nodes.

5.7 Performance evaluation

5.7.1 Simulated Environment and simulation settings

In this section we evaluate the performance of our offloading algorithm based on Reinforcement Learning algorithms. Namely, we present the performance of Actor Critic, Q-Learning with ε-greedy policy and Softmax policy. Performance evaluation is accomplished through a set of experiments in different scenarios. Performance results of our solution are compared with Droid, i.e., the benchmark solution we select for comparison – see Section 5.7.2 for a brief description.

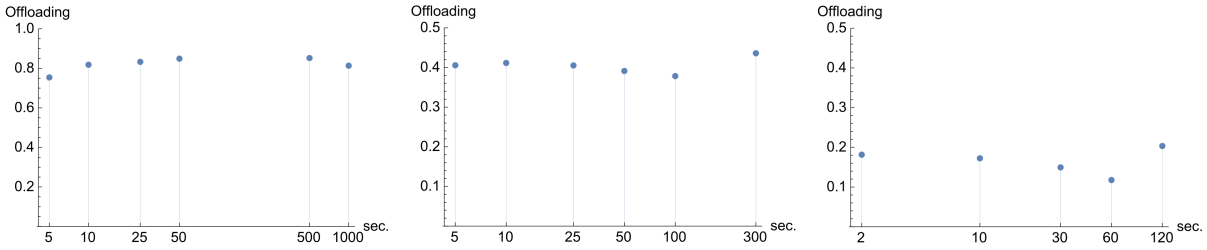
We consider a simulated environment with 600 mobile nodes moving in a $3000 \times 3000m^2$ area. Each mobile node represents a mobile device equipped with both Wifi and LTE network interfaces. The WiFi transmission range is set to 30m. Nodes mobility patterns are generated according to HCMM [25], a mobility model that integrates temporal, social and spatial notions in order to obtain an accurate representation of real user movements. Specifically, in HCMM the simulation space is divided in cells representing different social communities. Here we consider both a single community scenario (SC), a two-community scenario (MC2) and a five community scenario (MC5). For the single community scenario, all nodes percolate in the same area and have a chance to eventually meet all the other nodes in the area. In the multi community scenario we split the nodes in two and five separate communities located distant enough to each other such that no border effect may occur, i.e. node moving on the edge of two communities do not follow in the reciprocal transmission range. In this way, from the opportunistic network stand point, nodes belonging to different communities are not allowed to communicate to each other. In this setting we consider also the presence of special nodes that use to travel between communities, 10 for MC2 and 25 for MC5. They represent the only bridge between nodes of different communities, if we exclude the cellular network.

Algorithm 3 Offloading algorithm.

Let be alg the selected learning algorithm (AC or QL)
Let be c the requested content
Let be S_t the set of active seeds, S_t^c the rest of nodes waiting for the content c and S_{new} the number of new seeds.
Let be τ the elapsed time before the delivery time limit T.
Initialise $\boldsymbol{x}_0 = \{0, 0, 1\}$, $\boldsymbol{x}$, $p(\boldsymbol{x}, a) = 0\ \forall \boldsymbol{x}, a$
For each new content:
Draw the action a_0 from policy π for the state $\boldsymbol{x}_0$
$s = a_0 * |S_0^c|$
S_{new} ←GetNewSeeds(s)
Offload(S_{new})
$S_0 \leftarrow S_0 \cup S_{new}$
for each time step t **do**
    if $t \geq T$ **then**
        Transmit c to all nodes in S^c
        Stop content dissemination
    end if
    Measure $\boldsymbol{x}_t$, r_t
    UpdateRL(alg)
    Draw the action a_t from π_t for the state $\boldsymbol{x}_t$
    $s = a_t * |S_t^c|$
    S_{new} ←GetNewSeeds(s)
    Offload(S_{new})
    $S_t \leftarrow S_t \cup S_{new}$
end for
function GetNewSeeds(s)
    Sort nodes in S_t^c by "infection power"
    return the first s_{new} seeds
end function
function Offload(S_{new})
    Transmit c to all nodes in S_{new}
end function
function UpdateAC
    $\delta \leftarrow r_t + \gamma V^\pi(\boldsymbol{x}_t) - V^\pi(\boldsymbol{x}_{t-1})$
    $V^\pi(\boldsymbol{x}_{t-1}) \leftarrow V^\pi(\boldsymbol{x}_{t-1}) + \alpha\delta$
    $p(\boldsymbol{x}_{t-1}, a_{t-1}) \leftarrow p(\boldsymbol{x}_{t-1}, a_{t-1}) + \beta\delta$
end function
function UpdateQL
    $Q^\pi(\boldsymbol{x}_{t-1}, a_{t-1}) \leftarrow Q^\pi(\boldsymbol{x}_{t-1}, a_{t-1}) + \alpha \left[r_t + \gamma \max_a \hat{Q}^\pi(\boldsymbol{x}_t, a) - \hat{Q}^\pi_{old}(\boldsymbol{x}_{t-1}, a_{t-1}) \right]$
end function
function UpdateRL((alg))
    if alg is AC **then**
        UpdateAC()
    else
        UpdateQL()
    end if
end function

Notice that in the multi-community scenarios, nodes in each physical community move in a smaller area w.r.t. SC, therefore in MC2 and MC5 nodes more contact opportunities than in SC (although, clearly, the diversity of contacts for each node is lower). This aspect, as it will be clear in the following, if exploited well by an offloading algorithm, can have a significant impact on the offloading performance. Finally, all contact traces simulate one week of nodes' mobility.

In our experimental setup content to be delivered to interested nodes are generated sequentially. Each content must be delivered within a fixed deadline. We evaluated all the offloading approaches with the following content deadlines: $1000s$, $300s$, $120s$. During the life time of the content, in order to boost its dissemination, the controller can operate content re-injections every $5s$ for the $1000s$, $300s$ content deadlines and every $2s$ for the $120s$ content deadline. All results are averaged over 10 runs on 10 different contact traces, all one week long.



(a) 1000s, SC, 100% interested nodes (b) 300s, SC, 100% interested nodes (c) 120s, SC, 100% interested nodes

Figure 17: Droid parameter W sensitivity analysis. $C = 0.1$. Results for SC where all nodes are interested in receiving the content.

5.7.2 Droid summary

For the reader convenience, in this section we briefly describe the Droid's offloading strategy. In order to decide if new content replicas should be injected in the network, for each fixed time step, Droid performs the following actions. First, at time t it computes the slope of the dissemination ratio ($I(t)$) in the time window $[t - W, t]$ as discrete derivative:

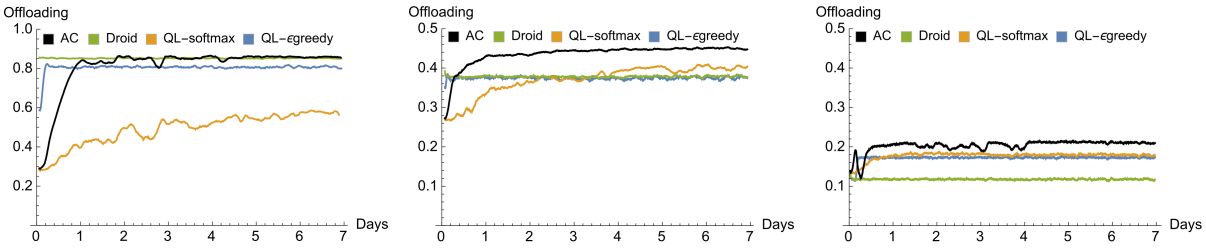
$$\Delta_I(t) = \frac{I(t) - I(t - W)}{W}$$

Second, Droid re-injects additional copies of the content if $\Delta_I(t)$ is below a threshold Δ_{lim} computed on line as the ratio between the fraction of nodes waiting the content and the time remaining (T) before the panic zone:

$$\Delta_{lim}(t) = \frac{1 - I(t)}{T}. \tag{11}$$

The new injection rate $r_{inj}(t)$ is computed as:

$$r_{inj}(t) = \begin{cases} c, & \Delta_I(t) \leq 0 \\ c(1 - \frac{\Delta_i(t)}{\Delta_{lim}(t)}), & 0 < \Delta_i(t) \leq \Delta_{lim}(t) \\ 0, & \Delta_I(t) > \Delta_{lim}(t). \end{cases}$$



(a) 1000s, SC, 100% interested nodes (b) 300s, SC, 100% interested nodes (c) 120s, SC, 100% interested nodes

Figure 18: Performance evaluation in single community with 100% of interested nodes. Considered content deadlines are $1000s$ $300s$ $120s$.

where $0 \leq c \leq 1$ is a clipping value used to limit the amount of content replicas to inject. Third, the number of new seeds $R(t)$ is computed as

$$R(t) = \lceil (1 - I(t)) \times |N(t)| \times r_{inj}(t) \rceil$$

where $|N(t)|$ is the number of nodes interested in the content. Finally, Droid selects $R(t)$ new nodes using a uniform distribution and offloads the content replicas.

In order to understand the results in Section 5.7.3, it is important to point out that the performance of Droid are strongly affected by the value of its parameters. As we can see form the above description, its behavior is controlled by the clipping value C and the width of the time window W. Among the two, we noticed by experiments that keeping fixed the clipping value, Droid performance is very sensitive the value of W. In Figure 17, we reported the mean offloading performance obtained in the SC scenario for all the content deadlines. On the $x-axis$ we have the time window W and on the y-axis the achieved offloading. Interestingly, in every considered scenario we notice that there exist a different value of W connected to the higher offloading result. This means that, in order to obtain the best result in every scenario we must tune the parameter W. In fact, a wrong choice of W can lead to very poor results.

5.7.3 Experimental Results

For the sake of simplicity from now on we will use the acronyms AC, Qe, Qs and DR for Actor Critic, Q-Learning with ε-greedy policy, Q-Learning with Softmax policy and Droid, respectively. To evaluate the performance of our approach, we observed and studied the evolution of the offloading level obtained with all the approaches. In light of what pointed out in Section 5.7.2, regarding Droid we report only its best performance for every scenario. The offloading level is defined as the percentage of nodes that received the content from the opportunistic network only. Moreover, in order to understand the internal behaviour of the considered offloading methods w.r.t. different scenarios, we analysed what actions are taken by the different algorithms.

5.7.4 Offloading performance

In order to understand which learning algorithm is more suitable for offloading we compared the performance of all the considered approaches. We start our analyses with a simple scenario. We consider distribution of content in a single community in which all nodes are interested in receiving it. Looking at Figure 18 we can notice that in absolute terms the offloading performance

Table 3: Levels of cellular offloading for SC, 100% of interested nodes.

Delivery thr.	AC	DR	Qe	Qs
1000s	85%	85%	80%	53%
300s	44%	37%	37%	40%
120s	20%	11%	17%	18%

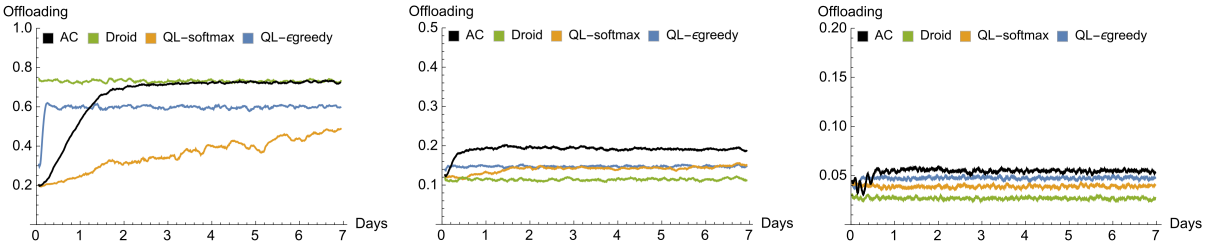
of all the approaches are strongly affected by two main factors: the nodes mobility expressed in terms of contact rate and the deadline for the delivery of a content. In fact, keeping fixed the contact rate, as long as we reduce the content deadline the absolute offloading level of all the approaches decreases accordingly. This means that for any scenario there is a limit for the amount of traffic that can be offloaded from the cellular network, and the objective of all the considered approaches is to regulate the controller behavior in order to approximate as much as possible that limit. As we can see from Figure 18(a), with a quite long deadline (w.r.t. the mobility considered), AC is able to offload, after almost 1 day spent to learn the best injection policy, as much traffic as the best fine tuned configuration of DR. Qe and Qs, instead, for different reasons show worse perform that AC and DR. Specifically, Qe thanks to the greedy policy has a steeper learning phase w.r.t AC, but following a greedy policy does not pay in the long term because after few time the learning phase get locked to a solution that is not the optimum. Conversely, Qs has a more explorative learning phase w.r.t. Qe but also very slow, leading to very poor performance.

For shorter deadlines (see Figure 18(b)-18(c)), the reaction of considered approaches are different, especially for DR Qe and Qe. AC proves to be able to well adapt its re-injection strategy in order to exploit as much as possible the combination of injected seeds and underlying mobility. DR proves to be less flexible to adapt its injection strategy, leading to poorer offloading performance. Qe and Qs still show worse offloading results w.r.t. AC but in this case they prove that being more reactive in terms of re-injections is beneficial to their offloading performance. Numerical results are reported in Table 3.

We performed a more challenging experiment in SC scenario where the total number of nodes interested in receiving the content is 20% of all the total number nodes, i.e., 120 out of 600 nodes. In this experiment, however, we allowed that the other nodes in the community to collaborate to the opportunistic dissemination process. In this experiment, seed nodes are always selected between interested nodes. This experimental setup represents a slightly difficult scenario w.r.t. the previous one, for all the offloading approaches because they must adapt their behaviour to the fact that the content can be delivered to interested nodes by other peers that are not directly controlled by the central controller. Therefore, here the challenge for the offloading algorithm is to understand that the opportunistic dissemination process could react with some delay to the content injection. Looking at Figure 19(a) we can notice AC is able to well adapt its injection policy and meet the best DR performance, offloading 72% of traffic. Qe and Qs, present a behavior very similar to the previous experiment leading to lower performance. Interestingly, for shorter deadlines we see the real benefit of AC. In fact, looking at Figures 19(b), 19(c), AC learns how to exploit better than Qe and Qs, what is the relation between the number of interested nodes, the contribution that can come from the other peers and the underlying mobility. This superior ability leads AC to offload more than the other approaches. Numerical

Table 4: Levels of cellular offloading for SC, 20% of interested nodes.

Delivery thr.	AC	DR	Qe	Qs
1000s	72%	72%	59%	40%
300s	19%	11%	14%	14%
120s	5%	2%	4%	3%



(a) 1000s, SC, 20% interested nodes (b) 300s, SC, 20% interested nodes (c) 120s, SC, 20% interested nodes

Figure 19: Offloading curves in SC scenario with 20% of interested nodes

results in Table 4.

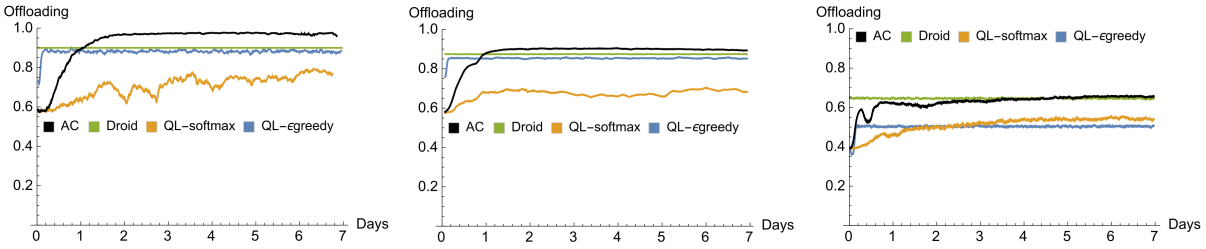
In order to push all these approaches to their limit we performed an even more challenging experiment in the SC scenario. As in the previous experiment, the number of nodes interested in receiving the content is 20% of the total number of nodes, but now all the other 80% of nodes do not contribute to the opportunistic dissemination process. Specifically, in this setting the central controller has to cope with the situation in which the effect of a content (re-)injection may arrive after quite long delay. In this case the offloading approaches have to find the right compromise between the number of seeds, the dissemination process and the time allowed before the deadline. As we can see from Table 5, only for 1000s deadline all the offloading approaches did produce results. In the other cases no content were offloaded. As in the former experiments, AC outperforms the other learning approaches and also Droid.

Now we show the performance of the offloading algorithms in two multi community scenarios. The motivation for this kind of experiments is to stress the approaches in different mobility scenarios. In fact, though we keep fixed the simulation area and mobility parameters, splitting node in 2 and 5 physical groups has the effect of increasing the contact rate between nodes. Therefore, in this scenarios it is expected to save more cellular traffic because of the more

Table 5: Levels of cellular offloading for SC, 20% of interested nodes. Non collaborative dissemination

Delivery thr.	AC	DR	Qe	Qs
1000s	33%	26%	18%	23%
300s	–	–	–	–
120s	–	–	–	–

contact possibilities that a seed or an infected node has to disseminate the content to other nodes. As expected, looking at Figure 20 and Tables 6-7, the overall offloading level for all the approaches and deadlines are very high. Apart from Qs that suffers of its slower learning ability, Qe, DR and AC are able to interpret very well the dynamics of the opportunistic scenario, reducing accordingly the number of seeds used to trigger the dissemination process.



(a) 1000s, MC, 100% interested nodes (b) 300s, MC, 100% interested nodes (c) 120s, MC, 100% interested nodes

Figure 20: Offloading curves in MC2 scenarion, 100% interested nodes

Table 6: Levels of cellular offloading for MC2, 100% of interested nodes.

Delivery thr.	AC	DR	Qe	Qs
1000s	97%	90%	88%	75%
300s	90%	87%	85%	69%
120s	65%	64%	50%	54%

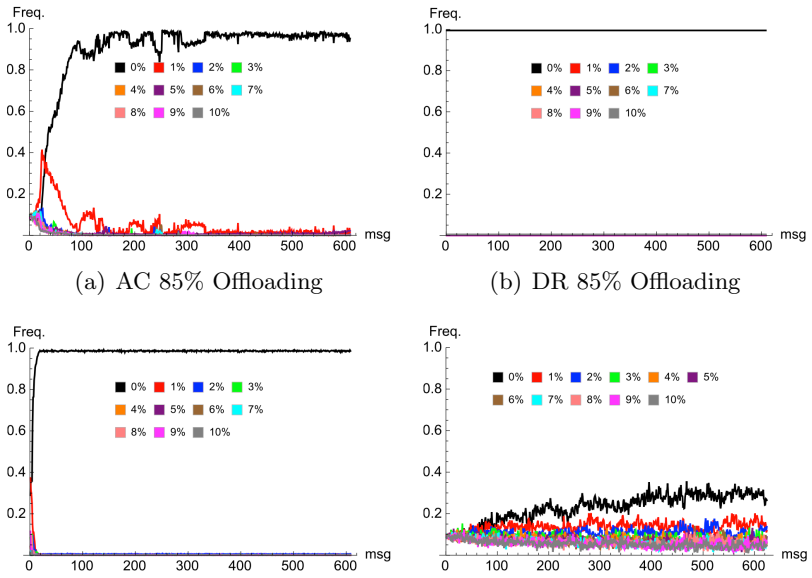
5.7.5 Analysis of injection policy

In order to understand the reason behind the just presented offloading performance, in this section we study the temporal evolution, content after content, of the (re-)injection actions performed by the central controller for all the considered approaches and content deadlines. More specifically, for each message sent during the one simulation run (1 week) we computed the relative frequencies of how many times each action has been taken by the controller. Then, we averaged results over 10 runs. For this analysis we focus on the single community scenario in which all nodes are interested in receiving the content.

In Figure 21 we collected the behavior of all the approaches corresponding to scenario with

Table 7: Levels of cellular offloading for MC5, 100% of interested nodes.

Delivery thr.	AC	DR	Qe	Qs
1000s	92%	89%	77%	73%
300s	95%	89%	77%	78%
120s	81%	81%	62%	68%



(a) AC 85% Offloading (b) DR 85% Offloading

(c) Qe 80% Offloading (d) Qs 53% Offloading

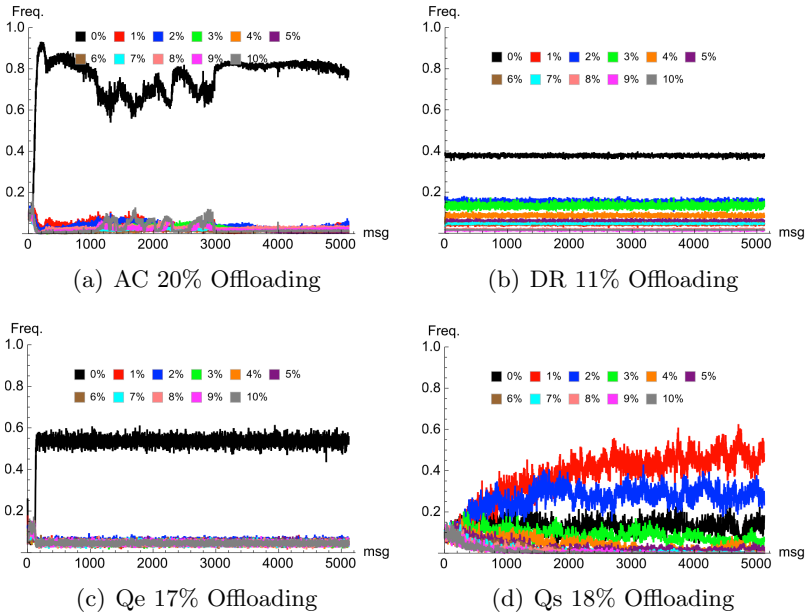
Figure 21: Actions frequencies curves. Deadline 1000s, SC, 100% interested nodes

1000s of content deadline. In that experiment, both AC and Droid offloaded 85% of traffic. Interestingly, this result is reflected in the type of actions taken by both approaches. Apparently, in this scenario the best strategy is to perform an initial injection and then do nothing form the rest of the time. In fact, DR does a first injection using 9% of seeds and no other injections do follow. AC shows the ability to learn that actually this is the best strategy, because if we look at the corresponding plot, after the learning phase we see that the most performed action is 0% of injection and the second most taken action is 1%. Also in the case of Qe, we see that in this scenario, given that there is enough time to let the opportunistic dissemination do the job, it is able to learn that the best policy is to do few little injections followed by no other actions for the rest of the time. Qs, instead, has a totally different behavior. As we can notice almost all the actions are taken with the same frequency and the use of the 0% action starts to increase sensitively just after 300 messages. This behavior is the direct consequence of the slow learning performance of QS.

Finally we performed the same analysis for content with 120s of delivery deadline. Such a tight deadline force the approaches to find a policy with a good balance between how aggressive should be the injections w.r.t. the dissemination capabilities of the opportunistic network. As reported in Figure 22 we see that the AC and Qe show similar a behavior: few more consistent injections (10%) followed by many 0% actions to permit the opportunistic dissemination do the rest. Droid due to the shorter deadline, adopted a very aggressive injection policy that resulted in just 11% of offloading performance.

5.8 Summary and outlook

We proposed a new solution to reduce the load of the network infrastructure based on Reinforcement Learning. Precisely, the infrastructure offloads part of the traffic to the underlying opportunistic network by selecting a subset of nodes to which delegate the dissemination of



(a) AC 20% Offloading

(b) DR 11% Offloading

(c) Qe 17% Offloading

(d) Qs 18% Offloading

Figure 22: Actions frequencies curves. Deadline 120s, SC, 100% interested nodes

contents in the Opportunistic Networks. In order to minimise the number of transmissions in charge of the cellular network we applied (separately) to learning algorithms, Actor Critic and Q-Learning, through which the controller learns, during time, what is the most appropriate injection policy. These learning approaches prove to have different performance and specifically the Actor Critic algorithm demonstrates a very good adaptability to different opportunistic scenarios, obtaining very high levels of offloading. In fact Compared with reference solutions in the literature, our approach in several configurations outperforms the benchmark solution in a number of configurations. Q-Learning based approaches, instead, though produce acceptable results, not always outperform Droid performance, the reference solution used to evaluate our approaches.

6 Security solutions for terminal-to-terminal communications

6.1 Integrity-preserving data dissemination in D2D networks

In this section, we study the problem of reliable communication in a multihop D2D network despite the presence of malicious attacks. The problem proves difficult since even a single malicious attacker node, if not neutralized, can lie to the entire network.

A common way to solve this problem is to use *cryptography* [26, 27]: the nodes use digital signatures to authenticate the sender across multiple hops. However, cryptography *per se* is not unconditionally reliable, as shown by the recent Heartbleed bug [28] discovered in the widely deployed OpenSSL software. The *defense in depth* paradigm [29] advocates the use of multiple layers of security controls, including non-cryptographic ones. For instance, if the cryptography-based security layer is compromised by a bug, a virus, or intentional tampering, a cryptography-free communication layer can be used to safely broadcast a patch or to update cryptographic keys. Thus, it is interesting to develop both cryptographic and non-cryptographic strategies.

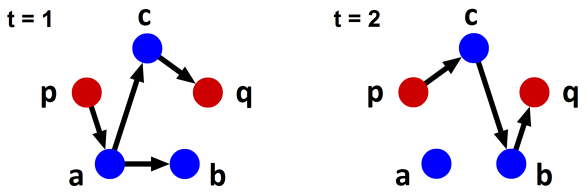


Figure 23: Counterexample to Menger's theorem in dynamic graphs.

In dense multihop networks, a first line of work assumes that there is a bound on the fraction of Byzantine nodes among the neighbors of each node. Protocols have been proposed for nodes organized on a grid [30, 31] (but with much more than 4 neighbors), and later generalized to other topologies [32], with the assumption that each node knows the global topology. Since this approach requires all nodes to have a large degree, it may not be suitable for every multihop networks. The case of sparse networks was studied under the assumption that malicious attackers occur uniformly at random [33, 34, 35], an assumption that holds, e.g., in structured overlay networks where the identifier (a.k.a. position) of a new node joining the network is assigned randomly, but not necessarily in various actual communication networks such as those envisioned in MOTO.

Most related to our work is the line of research that assume the existence of $2k + 1$ node-disjoint paths from source to destination, in order to provide reliable communication in the presence of up to k malicious attackers [36, 37]. The initial solution [36] assumes that each node is aware of the global network topology, but this hypothesis was dropped in subsequent work [37].

However, none of the aforementioned papers considers genuinely dynamic networks, i.e., where the topology evolves while the protocol executes. Some papers consider a dynamic model where the communication graph is complete [38], and some nodes can join and leave the network. Besides, they consider other problems than reliable communication. Another paper considers that the malicious attackers can affect different nodes over time [39]. Here, the graph is not fully connected, and the dynamicity is on the topology of the network.

In MOTO, our objective is to determine the condition for reliable communication in the presence of up to k malicious attackers in a *dynamic* network, where the topology can vary with time. The proof technique used in [36, 37] implicitly relies on Menger's theorem [40], which can be expressed as follows: there exists x disjoint paths between two nodes p and q if and only if x nodes must be removed to disconnect p and q.

However, Menger's theorem does not generalize to dynamic networks [41]. To illustrate this, let us consider the simple dynamic network of Figure 23. This network is in two steps ($t = 1$ and $t = 2$). There exists three dynamic paths connecting p to q: (p, a, c, q), (p, c, b, q) and (p, a, b, q). To cut these three paths, at least two nodes must be removed: either $\{a, b\}$, $\{b, c\}$ or $\{a, c\}$. Yet, it is impossible to find two disjoint paths among the three dynamic paths. Therefore, Menger's theorem cannot be used to prove the condition in dynamic networks, and the proof must be entirely remade.

We prove the necessary and sufficient condition for reliable communication in dynamic networks, in the presence of up to k malicious attackers. We consider the two cases where cryptography is available and not available. Our characterization is based on a dynamic version of

a minimal cut between p and q, denoted by DynMinCut(p, q), that takes into account both the presence of particular paths and their duration with respect to the delay that is necessary to actually transmit a message over a path. The condition is that DynMinCut(p, q) is lower or equal to $2k$ (without cryptography) or k (with cryptography). The proof is constructive, as we provide algorithms to prove the sufficiency of the condition.

In a second part, we apply these conditions to two case studies: participants interacting in a conference, and agents moving in the Paris subway. We thus show the benefit of this multihop approach for reliable communication, instead of waiting that the source meets the sink directly (if this event is to occur).

6.1.1 Preliminaries

Network model. We consider a continuous temporal domain $\mathbb{R}^+$ where dates are positive real numbers. We model the system as a time varying graph, as defined by Casteigts, Flocchini, Quattrociocchi and Santoro [42], where vertices represent the processes and edges represent the communication links (or channels). A time varying graph is a dynamic graph represented by a tuple $\mathcal{G} = (V, E, \rho, \zeta)$ where:

- V is the set of *nodes*.
- $E \subseteq V \times V$ is the set of *edges*.
- $\rho : E \times \mathbb{R}^+ \to \{0, 1\}$ is the *presence* function: $\rho(e, t) = 1$ indicates that edge e is present at date t.
- $\zeta : E \times \mathbb{R}^+ \to \mathbb{R}^+$ is the *latency* function: $\zeta(e, t) = T$ indicates that a message sent at date t takes T time units to cross edge e.

The discrete time model is a special case, where time and latency are restricted to integer values.

Hypotheses. We make the same hypotheses as previous work on the subject [30, 36, 31, 33, 35, 34, 37, 32]. First, each node has a unique identifier. Then, we assume *authenticated channels* (or *oral model*), that is, when a node q receives a message through channel (p, q), it knows the identity of p. Now, an omniscient adversary can select up to k nodes as *Byzantine*. These nodes can have a totally arbitrary and unpredictable behavior defined by the adversary (including tampering or dropping messages, or simply crashing). Finally, other nodes are *correct* and behave as specified by the algorithm. Of course, correct nodes are unable to know *a priori* which nodes are Byzantine. We also assume that a correct node u is aware of its *local topology* at any given date t (that is, u knows the set of nodes v such that $\rho((u, v), t) = 1$).

Dynamicity-related definitions. Informally, a *dynamic path* is a sequence of nodes a message can traverse, with respect to network dynamicity and latency.

Definition 1 (Dynamic path)**.** *A sequence of distinct nodes $(u_1, \ldots, u_n)$ is a dynamic path from u_1 to u_n if and only if there exists a sequence of dates $(t_1, \ldots, t_n)$ such that, $\forall i \in \{1, \ldots, n-1\}$ we have:*

- $e_i = (u_i, u_{i+1}) \in E$*, i.e. there exists an edge connecting u_i to u_{i+1}.*

- $\forall t \in [t_i, t_i + \zeta(e_i, t_i)]$, $\rho(e_i, t) = 1$, *i.e.* u_i *can send a message to* u_{i+1} *at date* t_i.
- $\zeta(e_i, t_i) \leq t_{i+1} - t_i$, *i.e. the aforementioned message is received by date* t_{i+1}.

We now define the *dynamic minimal cut* between two nodes p and q as the minimal number of nodes (besides p and q) one has to remove from the network to prevent the existence of a dynamic path between p and q. Formally:

- Let $Dyn\,(p, q)$ be the set of node sets $\{u_1, \ldots, u_n\}$ such that $(p, u_1, \ldots, u_n, q)$ is a dynamic path.
- For a set of node sets $\Omega = \{S_1, \ldots, S_n\}$, let $Cut\,(\Omega)$ be the set of node sets C such that, $\forall i \in \{1, \ldots, n\}$, $C \cap S_i \neq \emptyset$ (C contains at least one node from each set S_i).
- Let $MinCut\,(\Omega) = \min_{C\,Cut(\Omega)} |C|$ (the size of the smallest element of $Cut\,(\Omega)$). If $Cut(\Omega)$ is empty, we assume that $MinCut\,(\Omega) = +\infty$.
- Let $DynMinCut\,(p, q) = MinCut\,(Dyn\,(p, q))$.

Problem specification. We say that a node *multicasts* a message m when it sends m to all nodes in its current local topology. Now, a node u *accepts* a message m from another node v when it considers that v is the author of this message. We now define our problem specification, that is, *reliable* communication.

Definition 2 (Reliable communication)**.** *Let p and q be two correct nodes. An algorithm ensures reliable communication from p to q when the following two conditions are satisfied:*

- *When q accepts a message from p, p is necessarily the author of this message.*
- *When p sends a message, q eventually receives and accepts this message from p.*

6.1.2 Non-cryptographic reliable communication

In this section, we consider that cryptography is not available. We first provide a Byzantine-resilient multihop broadcast protocol. This algorithm is used as a constructive proof for the sufficient condition for reliable communication. We then prove the necessary and sufficient condition for reliable communication.

Informal description of the algorithm. Consider that each correct node p wants to broadcast a message m_0 to the rest of the network. Let us first discuss why the naive flood-based solution fails. A naive first idea would be to send a tuple (p, m_0) through all possible dynamic paths: thus, each node receiving m_0 knows that p broadcast m_0. Yet, Byzantine nodes may forward false messages, e.g., a Byzantine node could forward the tuple (p, m_1), with $m_1 \neq m_0$, to make the rest of the network believe that p broadcast m_1.

To prevent correct nodes from accepting a false message, we attach to each message the set of nodes that have been visited by this message since it was sent (that is, we use (p, m, S), where S is a set of nodes already visited by m since p sent it). As the Byzantine nodes can send any message, in particular, they can forward false tuples (p, m, S). Therefore, a correct node only accepts a message when it has been received through a collection of dynamic paths that cannot be cut by k nodes (where k is a parameter of the algorithm, and supposed to be an upper bound on the total number of Byzantine nodes in the network).

Variables. Each correct node u maintains the following variables:

- $u.m_0$, the message that u wants to broadcast.
- $u.\Omega$, a dynamic set registering all tuples (s, m, S) received by u.
- $u.Acc$, a dynamic set of confirmed tuples (s, m). We assume that whenever $(s, m) \in u.Acc$, u accepts m from s.

Initially, $u.\Omega = \{(u, u.m_0, \emptyset)\}$ and $u.Acc = \{(u, u.m_0)\}$.

Algorithm. Each correct node u obeys the three following rules:

1. Initially, and whenever $u.\Omega$ or the local topology of u change: multicast $u.\Omega$.
2. Upon reception of Ω' through channel (v, u): $\forall (s, m, S) \in \Omega'$, if $v \notin S$ then append $(s, m, S \cup \{v\})$ to $u.\Omega$.
3. Whenever there exist s, m and $\{S_1, \ldots, S_n\}$ such that $\forall i \in \{1, \ldots, n\}$, $(s, m, S_i \cup \{s\}) \in u.\Omega$ and $MinCut\,(\{S_1, \ldots, S_n\}) > k$: append (s, m) to $u.Acc$.

Condition for reliable communication. Let us consider a given dynamic graph, and two given correct nodes p and q. Our main result is as follows:

Theorem 1. *For a given dynamic graph, a k-Byzantine tolerant reliable communication from p to q is feasible if and only if $DynMinCut\,(p, q) > 2k$.*

Proof. The proof of the "only if" part is in Lemma 1. The proof of the "if" is in Lemma 4. □

Lemma 1 (Necessary condition)**.** *For a given dynamic graph, let us suppose that there exists an algorithm ensuring reliable communication from p to q. Then, we necessarily have $DynMinCut\,(p, q) > 2k$.*

Proof. Let us suppose the opposite: there exists an algorithm ensuring reliable communication from p to q, and yet, $DynMinCut\,(p, q) \leq 2k$. Let us show that it leads to a contradiction.

As we have

$$DynMinCut\,(p, q) = MinCut\,(Dyn\,(p, q)) \leq 2k$$

and

$$MinCut\,(Dyn\,(p, q)) = \min_{C \in Cut(Dyn(p,q))} |C|$$

there exists an element C of $Cut\,(Dyn\,(p, q))$ such that $|C| \leq 2k$. Let C_1 be a subset of C containing k' elements, with $k' = \min(k, |C|)$. Let $C_2 = C - C_1$. Thus, we have $|C_1| \leq k$ and $|C_2| \leq k$.

According to the definition of $Cut\,(Dyn\,(p, q))$, C contains a node of each possible dynamic path from p to q. Therefore, the information that q receives about p are completely determined by the behavior of the nodes in C.

Let us consider two possible placements of Byzantine nodes, and show that they lead to a contradiction:

- First, suppose that all nodes in C_1 are Byzantine, and that all other nodes are correct. This is possible since $|C_1| \leq k$.

  Suppose now that p broadcasts a message m. Then, according to our hypothesis, since the algorithm ensures reliable communication, q eventually accepts m from p, regardless of what the behavior of the nodes in C_1 may be.

- Now, suppose that all nodes in C_2 are Byzantine, and that all other nodes are correct. This is also possible since $|C_2| \leq k$.

  Then, suppose that p broadcasts a message $m' \neq m$, and that the Byzantine nodes have exactly the same behavior as the nodes of C_2 had in the previous case.

  Thus, as the information that q receives about p is completely determined by the behavior of the nodes of C, from the point of view of q, this situation is indistinguishable from the previous one: the nodes of C_2 have the same behavior, and the behavior of the nodes of C_1 is unimportant. Thus, similarly to the previous case, q eventually accepts m from p.

Therefore, in the second situation, p broadcasts m, and q eventually accepts $m' \neq m$. Thus, according to Definition 2, the algorithm does not ensure reliable communication, which contradicts our initial hypothesis. Hence, the result. □

Lemma 2 (Safety)**.** *Let us suppose that all correct nodes follow our algorithm. If* $(p, m) \in q.Acc$*, then* $m = p.m_0$*.*

Proof. As $(p, m) \in q.Acc$, according to rule 3 of our algorithm, there exists $\{S_1, \ldots, S_n\}$ such that, $\forall i \in \{1, \ldots, n\}$, $(p, m, S_i \cup \{p\}) \in q.\Omega$, and $MinCut\left(\{S_1, \ldots, S_n\}\right) > k$.

Suppose that each node set $S \in \{S_1, \ldots, S_n\}$ contains at least one Byzantine node. If C is the set of Byzantine nodes, then $C \in Cut\left(\{S_1, \ldots, S_n\}\right)$ and $|C| \leq k$. This is impossible because $MinCut\left(\{S_1, \ldots, S_n\}\right) > k$. Therefore, there exists $S \in \{S_1, \ldots, S_n\}$ such that S does not contain any Byzantine node.

Now, let us use the correct dynamic path corresponding to S to show that $m = m_0$. Let $n' = |S \cup \{p\}|$. Let us show the following property $\mathcal{P}_i$ by induction, $\forall i \in \{0, \ldots, n'\}$: there exists a correct node u_i and a set of correct nodes X_i such that $(p, m, X_i) \in u_i.\Omega$ and $|X_i| = |S \cup \{p\}| - i$.

- As $S \in \{S_1, \ldots, S_n\}$, $(p, m, S \cup \{p\}) \in q.\Omega$. Thus, $\mathcal{P}_0$ is true if we take $u_0 = q$ and $X_0 = S \cup \{p\}$.

- Let us now suppose that $\mathcal{P}_{i+1}$ is true, for $i < n'$. As $(p, m, X_i) \in u_i.\Omega$, according to rule 2 of our algorithm, it implies that u_i received Ω' from a node v, with $(p, m, X) \in \Omega'$, $v \notin X$ and $X_i = X \cup \{v\}$. Thus, $|X| = |X_i| - 1 = |S \cup \{p\}| - (i + 1)$.

  As $v \in X_i$ and X_i is a set of correct nodes, v is correct and behaves according to our algorithm. Then, as v sent Ω', according to rule 1 of our algorithm, we necessarily have $\Omega' \subseteq v.\Omega$. Thus, as $(p, m, X) \in \Omega'$, we have $(p, m, X) \in v.\Omega$. Hence, $\mathcal{P}_{i+1}$ is true if we take $u_{i+1} = v$ and $X_{i+1} = X$.

By the induction principle, $\mathcal{P}_{n'}$ is true. As $|X_{n'}| = 0$, $X_{n'} = \emptyset$ and $(p, m, \emptyset) \in u_{n'}$. As $u_{n'}$ is a correct node and follows our algorithm, the only possibility to have $(p, m, \emptyset) \in u_{n'}.\Omega$ is that $u_{n'} = p$ and $m = p.m_0$. Thus, the result. □

Lemma 3 (Communication)**.** *Let us suppose that DynMinCut* $(p, q) > 2k$*, and that all correct nodes follow our algorithm. Then, we eventually have* $(p, p.m_0) \in q.Acc$*.*

Proof. Let $\{S_1, \ldots, S_n\}$ be the set of node sets $S \in Dyn\,(p, q)$ that only contain correct nodes. Similarly, let $\{X_1, \ldots, X_{n'}\}$ be the set of node sets $X \in Dyn\,(p, q)$ that contain at least one Byzantine node.

Let us suppose that $MinCut\,(\{S_1, \ldots, S_n\}) \leq k$. Then, there exists $C \in Cut\,(\{S_1, \ldots, S_n\})$ such that $|C| \leq k$. Let C' be the set containing the nodes of C and the Byzantine nodes. Thus, and $C' \in Cut\,(\{S_1, \ldots, S_n\} \cup \{X_1, \ldots, X_{n'}\}) = Cut\,(Dyn\,(p, q))$, and $|C'| \leq 2k$. Thus, $MinCut\,(Dyn\,(p, q)) \leq 2k$, which contradicts our hypothesis. Therefore, $MinCut\,(\{S_1, \ldots, S_n\}) > k$.

In the following, we show that $\forall S \in \{S_1, \ldots, S_n\}$, we eventually have $(p, p.m_0, S \cup \{p\}) \in q.\Omega$, ensuring that q eventually accepts $p.m_0$ from p.

Let $S \in \{S_1, \ldots, S_n\}$. As $S \in Dyn\,(p, q)$, let $(u_1, \ldots, u_N)$ be the dynamic path such that $p = u_1$, $q = u_N$ and $S = \{u_2, \ldots, u_{N-1}\}$. Let $(t_1, \ldots, t_N)$ be the corresponding dates, according to Definition 1. Let us show the following property $\mathcal{P}_i$ by induction, $\forall i \in \{1, \ldots, N\}$: at date t_i, $(p, p.m_0, X_i) \in u_i.\Omega$, with $X_i = \emptyset$ if $i = 1$ and $\{u_1, \ldots, u_{i-1}\}$ otherwise.

- $\mathcal{P}_1$ is true, as we initially have $(p, p.m_0, \emptyset) \in p.\Omega$.
- Let us suppose that $\mathcal{P}_i$ is true, for $i < N$. According to Definition 1, $\forall t \in [t_i, t_i + \zeta(t_i, u_i)]$, $\rho(e_i, t) = 1$, e_i being the edge connecting u_i to u_{i+1}.
  - Let $t_A \leq t_i$ be the earliest date such that, $\forall t \in [t_A, t_i + \zeta(t_i, u_i)]$, $\rho(e_i, t) = 1$.
  - Let $t_B \leq t_i$ be the date where (p, m, X_i) is added to $u_i.\Omega$.
  - Let $t_C = max(t_A, t_B)$.

  Then, at date t_C, either $u_i.\Omega$ or the local topology topology of u_i changes. Thus, according to rule 1 of our algorithm, u_i multicasts $\Omega' = u_i.\Omega$ at date t_C, with $(p, p.m_0, X_i) \in \Omega'$.

  As $\zeta(e_i, t_i) \leq t_{i+1} - t_i \leq t_{i+1} - t_C$, u_{i+1} receives Ω' from u_i at date $t_C + \zeta(e_i, t_i) \leq t_{i+1}$. Then, according to rule 2 of our algorithm, $(p, p.m_0, X_i \cup \{u_i\})$ is added to $u_{i+1}.\Omega$.

  Thus, $\mathcal{P}_{i+1}$ is true if we take $X_{i+1} = X_i \cup \{u_i\}$.

By induction principle, $\mathcal{P}_N$ is true. As $u_1 = p$, $X_N = \{u_1, \ldots, u_{N-1}\} = S \cup \{p\}$, and we eventually have $(p, p.m_0, S \cup \{p\}) \in q.\Omega$.

Thus, $\forall S \in \{S_1, \ldots, S_n\}$, we eventually have $(p, p.m_0, S \cup \{p\}) \in q.\Omega$. Then, as

$$MinCut\,(\{S_1, \ldots, S_n\}) > k$$

according to rule 3 of our algorithm, $(p, p.m_0)$ is added to $q.Acc$. □

Lemma 4 (Sufficient condition)**.** *Let there be any dynamic graph. Let* p *and* q *be two correct nodes, and* k *denote the maximum number of Byzantine nodes. If DynMinCut* $(p, q) > 2k$*, our algorithm ensures reliable communication from* p *to* q*.*

Proof. Let us suppose that the correct nodes follow our algorithm, as described in Section 6.1.2. First, according to Lemma 2, if $(p, m) \in q.Acc$, then $m = p.m_0$. Thus, when q accepts a message from p, p is necessarily the author of this message. Then, according to Lemma 3, we eventually have $(p, p.m_0) \in q.Acc$. Thus, q eventually receives and accepts the message broadcast by p. Therefore, according to Definition 2, our algorithm ensures reliable communication from p to q. □

6.1.3 Cryptographic reliable communication

If cryptography is available, then it becomes possible to authenticate the sender of a message across multiple hops.

The setting is now the following. Each node p has a private key $priv_p$ (only known by p) and a public key pub_p (known by all nodes). The node p can encrypt a message m with the function $crypt(priv_p, m)$. Any node q can decrypt a message from p with the function $decrypt(pub_p, m)$. This function returns NULL if the message was not correctly encrypted. We assume that the Byzantine nodes do not know the private keys of correct nodes.

Then, we modify the previous algorithm as follows. Initially, $u.\Omega = \{(u, crypt(priv_u, u.m_0))\}$ and $u.Acc = \{(u, u.m_0)\}$. Then, each correct node u obeys to the three following rules:

1. Initially, and whenever $u.\Omega$ or the local topology of u change: multicast $u.\Omega$.
2. Upon reception of Ω' from a neighbor node: $u.\Omega = u.\Omega \cup \Omega'$.
3. Whenever there exists $(s, m) \in u.\Omega$ such that $m' = decrypt(pub_s, m) \neq NULL$: append (s, m') to $u.Acc$.

Theorem 2. *If cryptography is available, for a given dynamic graph, a k-Byzantine tolerant reliable communication from p to q is feasible if and only if $DynMinCut\,(p, q) > k$.*

Proof. If $DynMinCut\,(p, q) \leq k$, then it is possible to cut all dynamic paths between p and q with Byzantine nodes. Thus, q never receives any message from p. Thus, the condition is necessary. Now, let us show that the condition is sufficient.

First, q cannot accept a message (p, m) with $m \neq p.m_0$. Indeed, let us suppose the opposite. According to step 3 of the algorithm, it implies that we have $(p, m') \in q.\Omega$, with $decrypt(pub_p, m') = m$. Implying that $m' = crypt(priv_p, m)$. Let v be the first node to have $(p, m') \in v.\Omega$. According to steps 1 and 2 of the algorithm, v cannot be a correct node. Thus, v is Byzantine, implying that a Byzantine node knows $priv_p$: contradiction.

Besides, if $DynMinCut\,(p, q) > k$, then there exists at least one dynamic path from p to q. Thus, for the same argument as in Lemma 3, we eventually have $(p, crypt(priv_p, p.m_0)) \in q.\Omega$. Thus, according to step 3 of the algorithm, $(p, p.m_0)$ is added to $q.Acc$, and the condition is sufficient.

□

6.1.4 Case Studies

In this section, we apply our conditions for reliable communication to several case studies: participants interacting in a conference, and agents moving in the subway. We show the interest of multihop reliable communication.

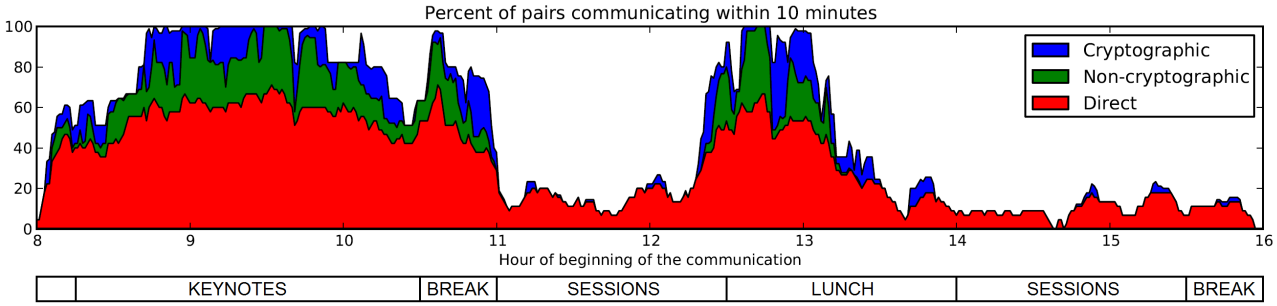


Figure 24: Reliable communication between 10 most sociable nodes of the Infocom 2005 dataset

A real-life dynamic network: the Infocom 2005 dataset. In this section, we consider the Infocom 2005 dataset [43], which is obtained in a conference scenario by iMotes capturing contacts between participants. This dataset can represent a dynamic network where each participant is a node and where each contact is a (temporal) edge.

We consider an 8-hour period during the second day of the conference. In this period, we consider the dynamic network formed by the 10 most "sociable" nodes (our criteria of sociability is the total number of contacts reported). We assume that at most one on these nodes may be Byzantine (that is, $k = 1$).

Let p and q be two correct nodes. Let us suppose that p wants to transmit a message to q within a period of 10 minutes. Within 10 minutes, three types of communication can be achieved:

- *Direct* communication: p meets q directly.
- *Non-cryptographic* communication: the condition for reliable non-cryptographic communication (Theorem 1) is satisfied.
- *Cryptographic* communication: the condition for reliable cryptographic communication (Theorem 2) is satisfied.

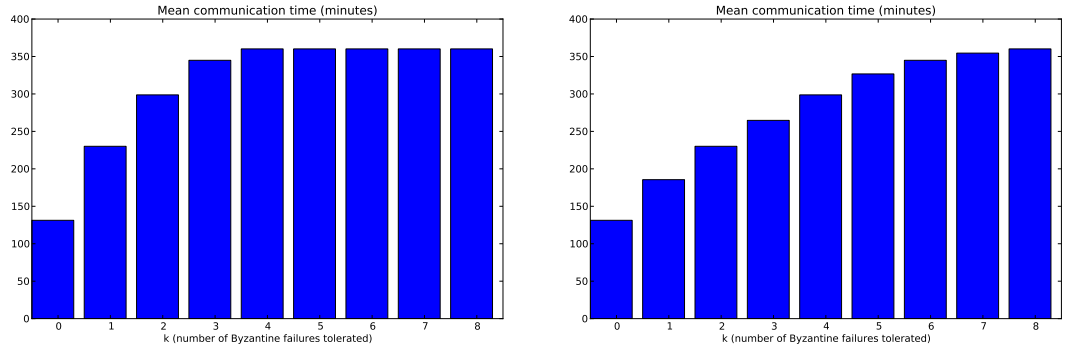
If we want to ensure reliable communication despite one Byzantine node, the simplest strategy is to wait until p meets q directly. Let us show now that relaying the message is usually beneficial and that our approach realizes a significant gain of performance.

Figure 24 represents the percentage of pairs of nodes (p, q) that communicate within 10 minutes, according to the date of beginning of the communication. We can correlate the peaks with the program of the conference: the first period corresponds to morning arrivals during the keynotes; the peak between 10:30 and 11:00 corresponds to the morning break; the peak starting at 12:30 corresponds to the end of parallel sessions and the departure for lunch.

As it turns out, many pairs of nodes are able to communicate reliably, even though they are unable to meet directly. For instance, at 9:15, 60% of pairs of nodes meet directly, 80% can communicate reliably without cryptography, and 100% can communicate reliably with cryptography. This means that relaying the information is actually effective and desirable.

Mobile agents in the Paris subway. We consider a dynamic network consisting of 10 mobile agents randomly moving in the Paris subway. The agents can use the classical subway lines (we exclude tramways and regional trains). Each agent is initially located at a randomly chosen junction station – that is, a station that connects at least two lines. Then, the agent randomly chooses a neighbor junction station, waits for the next train, moves to this station, and repeats the process. We use the train schedule provided by the local subway company (`http://data.ratp.fr`). The time is given in minutes from the departure of the first train (i.e., around 5:30). We consider that two agents can communicate in the two following cases:

1. They are staying together at the same station.
2. They cross each other in trains. For instance, if at a given time, one agent is in a train moving from station A to station B while the other agent moves from B to A, then we consider that they can communicate.



(a) Mean communication time without cryptography (subway)

(b) Mean communication time with cryptography (subway)

Figure 25: Mean communication time in the Paris subway

Similarly to the previous case study, we represented the mean communication time with and without cryptography (see Figure 25(a) and 25(b)). The qualitative observations are the same.

Again, let us suppose that we want to tolerate one Byzantine failure ($k = 1$). Let us consider the mean time for p and q to meet directly. If we use our algorithms, this time decreases by 36% without cryptography, and by 49% with cryptography.

6.2 MOTO security solution for D2D communications

As it can be concluded from the previous sections, reliable communications in opportunistic networks are feasible without the use of encryption. However, attending to efficiency, the use of cryptography provides better network performance (reducing the overload) if the integrity of the content transmitted needs to be guaranteed. On the other hand, assuring only integrity is considered insufficient. Content must be disclosed only to its intended recipients, and this condition can only be assured with the use of a robust encryption approach. For these reasons, integrity, confidentiality and encryption are considered necessary for terminal to terminal communications under the MOTO proposed approach.

6.2.1 Preliminary conditions

In order to set up the security protocol that will guide the opportunistic communications in MOTO, some assumptions are considered:

- Node behavior is unpredictable. This means that all nodes are susceptible of connecting and disconnecting at any moment, as well as acting in a malicious or selfish way.
- Nodes resources are scarce. Participating nodes are mainly mobile handsets with limited battery life and processing capabilities.
- MOTO platform is always accessible for the nodes. All nodes enjoying MOTO services can communicate with the MOTO platform through a reliable connection when required.
- The content to be delivered through MOTO services is encrypted in the origin (Content provider), and intended recipients are in possession of the required decryption key.
- All authorized MOTO users are in possession of the public key of the MOTO platform, while the private encryption key of the MOTO platform is only known by itself.
- All authorized users possess static asymmetric encryption keys, only known by them and by the MOTO platform, securely stored in the device, and no other node knows to which node corresponds each encryption key.
- The MOTO platform connection with the nodes is considered reliable.
- The MOTO platform cannot be impersonated.
- All authorized nodes of MOTO services are provided periodically a set of public and private key pairs that are correlated with temporary pseudonyms (temporal ID from which the real identity of the nodes cannot be deduced).

Taking all these assumptions into account, the security protocol for terminal to terminal communication proposed guarantees the following aspects:

- Reliable transmission of content through hop to hop communications. Integrity and confidentiality of the content is assured through the opportunistic network.
- Detection and suspension of authenticated user acting selfishly or in a malicious manner.
- Identity and location information preservation of all nodes. The real identity and the location of all participating nodes must remain secret to all other nodes, and only will be available for the MOTO platform.

6.2.2 Protocol Notation

To expose the security protocol the notation used in [44] is taken into account. In this notation, the communications of the protocol are numbered with the label Message 1, Message 2, etc., while T represents a timestamp, N a nonce (a random value generated to check that a message is recent), K represents an encryption Key, and K^{-1} its inverse. Finally, a session encryption key (symmetric key) between two nodes A and B will be represented as K_{ab}. However, the

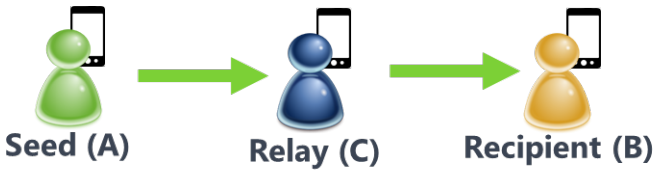


Figure 26: Sample Opportunistic Network

notation adopted for public and private keys is different, and K_{pux} is used for the static public encryption key and K_{prx} of a user x (which will only be used when communicating with the MOTO platform). When communicating with other opportunistic nodes, K_{puxi} is used for the public part of a key pair (public Key) and K_{prxi} for the secret (private) part of the key pair correlated to Pseudonym i of a node x. Content to be delivered is represented with C, and the encryption of a content C using a key K is represented with $\{C\}_k$. Finally, the pseudonyms are represented as P_{xi}, where x represents the node that holds the pseudonym in that instance while i represents the numbering of the pseudonyms related to all the pseudonyms in possession of node x. Using this notation, the first message of a protocol in which a node A sends a timestamp, and an encrypted content using its private key correlated to its first pseudonym, to a user B is represented:

Message 1 $A \rightarrow B : T\{C\}K_{pra1}$

6.2.3 Pseudonyms distribution Protocol

Consider the minimum set of nodes that capture a full sampling of the MOTO proposed opportunistic approach. A seed node A is in possession of a content M that is intended for node B (Recipient). Consider that node A cannot establish a direct communication with node B, and therefore needs the assistance of a relay node (C). The proposed sample scenario is shown in Figure 26.

As already exposed, all nodes A, B, and C (real user's identity) can reach the MOTO platform (M) through a reliable communication. First, each of the participating nodes has been provided a set of public-private key pairs and the corresponding pseudonyms. Consider that N is a connection request message (Nonce) specific for each node. The first messages of the protocol are:

Message 1 $M \rightarrow A : \{N_1, T_1\}K_{pua}$
Message 1' $M \rightarrow B : \{N_2, T_2\}K_{pub}$
Message 1" $M \rightarrow C : \{N_3, T_3\}K_{puc}$
Message 2 $A \rightarrow M : \{A, N_1\}K_{puM}$
Message 2' $B \rightarrow M : \{B, N_2\}K_{puM}$
Message 2" $C \rightarrow M : \{C, N_3\}K_{puM}$
Message 3 $M \rightarrow A : \{P_{a1}, P_{a2}, P_{a3}, K_{pra1}, K_{pra2}, K_{pra3}, K_{pua1}, K_{pua2}, K_{pua3}\}K_{pua}$
Message 3' $M \rightarrow B : \{P_{b1}, P_{b2}, P_{b3}, K_{prb1}, K_{prb2}, K_{prb1}, K_{pub1}, K_{pub2}, K_{pua3}\}K_{pub}$
Message 3" $M \rightarrow C : \{P_{c1}, P_{c2}, P_{c3}, K_{prb1}, K_{prb2}, K_{prb3}, K_{pub1}, K_{pub2}, K_{pub3}\}K_{puc}$
Message 4 $A \rightarrow M : \{A, N_1, P_{a1}, P_{a2}, P_{a3}, T_4\}K_{puM}$
Message 4' $B \rightarrow M : \{B, N_2, P_{b1}, P_{b2}, P_{b3}, T_5\}K_{puM}$
Message 4" $C \rightarrow M : \{C, N_3, P_{c1}, P_{c2}, P_{c3}, T_6\}K_{puM}$

Figure 27: Proposed contact for the security protocol

Once these messages have been exchanged, the MOTO platform can assure that the user has received and updated the pseudonyms and their correlated key pairs, and will update its database to reflect it. This protocol grants the following aspects:

1. Each of the Nonce delivered by the MOTO platform in messages 1, 1' and 1", are used to assure that the answer received in messages 2, 2' and 2" are from the nodes intended.

2. The timestamp from the messages 1, 1' and 1" inform about when the MOTO platform starts the process of pseudonym updating.

3. The nodes Id in messages 2, 2' and 2" allow the confirmation of the validity of the message, correlation between nonce and user Identity.

4. Finally, sending user identity along with the pseudonyms received in messages 4, 4' and 4", allows the MOTO platform to validate the correct reception of the pseudonyms for each node.

5. The timestamps allow to know from which moment the pseudonyms are valid for each node. This way, the timestamps of messages 4, 4' and 4" state when the nodes A, B and C will start using the new pseudonyms.

6.2.4 Content transmission protocol

This part of the Security Protocol deals with the content delivery in each contact opportunity. The steps defined are independent of the type of nodes involved in the communication (seed, relay or receipt), as the same steps will be applicable for all of the contacts made in the opportunistic network. Let's Consider that a node A receives a content C with Hash HC from B (1 Hop).

The aim of the protocol is to provide a secure transmission of the content through the connectivity capabilities of the nodes, without revealing the real identity of the users, or disclosing the content (C) to other nodes. The protocol must also assure that users are reliable and belong to the MOTO services. The proposed steps of the protocol to deal with the delivery of this content from B to A will be:

Message 1 $A \rightarrow B : P_{a1}$
Message 2 $B \rightarrow A : P_{b1}$
Message 3 $B \rightarrow M : \{P_{a1}, B\}K_{puM}$
Message 3' $A \rightarrow M : \{P_{b1}, A\}K_{puM}$
Message 4 $M \rightarrow B : \{K_{pua1}, N_1, N_2\}K_{pub}$
Message 4 $M \rightarrow A : \{K_{pub1}, N_1, N_2\}K_{pua}$

Message 5 $B \rightarrow A : \{N_1, K_{ab}\}K_{pua1}$
Message 6 $A \rightarrow B : \{N_2, H_c, K_{ab}\}K_{pub}$
Message 7 $B \rightarrow A : \{C\}K_{ab}$

With the proposed protocol, users avoid to disclose to each other their real identity and rely in the MOTO platform to distribute the public part of the key which corresponds to the given pseudonym. Between Steps 3 and 4 of the protocol, the MOTO platform will check each user's permissions to participate in the MOTO services (Trust), and will act consequently. Note that in the described sequence, both users have been found trustable. The protocol grants the following aspects:

1. The use of Pseudonyms between nodes (message 1 and 2) avoids the disclosure of node real identity between the nodes.

2. Messages 3 and 4 are the request that users A and B do to the MOTO platform in order to receive the public key of the other node, this request also gives the opportunity to receive instructions from the MOTO platform about the trustworthiness of the nodes. Moto platform will dispatch, only if the user's trust is above threshold, a public key which corresponds to the pseudonym sent. MOTO platform will also include in the answer two nonce values aimed to help both nodes to assure each other. These nonce assist the nodes to validate each other's trustworthiness.

6.2.5 Feedback transmission protocol

Another relevant process involved in the MOTO communication scheme is the submission of the feedback regarding the communication. This feedback is transmitted by the each node involved in a communication through the opportunistic network. The feedback must accomplish the following objectives:

1. The origin of the feedback must be guaranteed; no malicious user should be able to get a false feedback taken as a valid one.

2. The identity of the user sending the feedback must be kept secret; no other node should be able to identify the real identity of the user sending the feedback.

3. The feedback must only be readable by the MOTO platform.

4. The feedback integrity must be assured.

To set the steps of the protocol, a scenario where one user sends a feedback regarding a communication that has taken place is sufficient. Let's take a user A which has just received a piece of content from another user, user B. The Hash of the received content will be HC and the pseudonyms of user A and User B will be P_{a1} and P_{b1} respectively. Finally, the feedback is notated as FAB. *Note that only the MOTO platform is in possession of User's A public key.*

In order to accomplish these objectives, the steps stated for the protocol are as follow:

Message 1 $A \rightarrow M : \{P_{a1}, N_1\}K_{puM}$
Message 2 $M \rightarrow A : \{N_1, N_2\}K_{pua}$
Message 3 $A \rightarrow M : \{N_2, FAB, HC, TAB, P_{a1}, P_{b1}\}K_{puM}$

The steps of the protocol guarantee the objectives stated previously, as:

- The nonce (N_1) is used by user A to assure that only the MOTO platform will create the answer used in message 2.
- The second nonce (N_2) is created by the MOTO platform to assure that user A is really who it claims to be when it is included in message 3.

6.3 Summary and outlook

The security protocol that has been defined in the previous sections provides the aimed objectives for the security assurance in the opportunistic network of the MOTO communications approach. The steps included allow the secure distribution of pseudonyms and correlated encryption key pairs, content distribution and feedback delivery. The protocol has been analyzed under different attacks, such as Man in the Middle, and it has till now demonstrated to be robust against malicious attempts against content confidentiality and integrity and user identity disclosure. Finally, the protocol also provides assurance regarding the origin of the communications.

7 Conclusion

We have presented in this deliverable two main contributions related to terminal-to-terminal communications protocols. We proposed EPICS, a generic and extensible distributed strategy based on the grey relational analysis for inter-content piece selection when two nodes observe a contact opportunity. We evaluated the performance of EPICS in our testbed composed by 10 Android smartphones and showed that, when compared to a uniform strategy, EPICS ensures fairer dissemination delays for all the contents regardless of their creation times and sizes. We have also evaluated EPICS in a scenario with frequent contacts and inter-contacts. As a follow-up of EPICS, we proposed DAD, a solution that considers the convenience of transmitting a burst of pieces at each opportunistic contact. Results show the need for dynamic burst dimensioning based on node degree.

We have also proposed a new solution to reduce the load of the network infrastructure. Precisely, the infrastructure offloads part of the traffic to the underlying opportunistic network by selecting a subset of nodes to which delegate the dissemination of contents in the Opportunistic Networks. In order to minimise the number of transmissions in charge of the cellular network we applied an Actor Critic technique through which the controller learns, during time, what is the most appropriate injection policy. Our solution demonstrates a very good adaptability to different opportunistic scenarios, obtaining very high levels of offloading.

Finally, we addresses security issues with regard to opportunistic communications. Firstly, we proved the necessary and sufficient condition for reliable communication in opportunistic networks in the presence of up to k malicious attackers. We considered the two cases where cryptography is available and not available. Our characterization is based on a dynamic version of a minimal cut between p and q that takes into account both the presence of particular paths and their duration with respect to the delay that is necessary to actually transmit a message over a path. Secondly, we defined a security protocol that covers the objectives for the security assurance in the opportunistic network of the MOTO communications approach. The security aspects are to be associated with deliverable D4.3 (*Trust and security issues and solution*).

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2013–2018," White Paper, Cisco, Feb. 2014.

[2] N. Belblidia, M. D. de Amorim, L. H. M. K. Costa, J. Leguay, and V. Conan, "Part-whole dissemination of large multimedia contents in opportunistic networks," *Computer Communications*, vol. 35, no. 15, pp. 1786–1797, Mar. 2012.

[3] J. L. Deng, "Control problems of grey systems," *Systems & Control Letters*, vol. 1, no. 5, pp. 288–294, Mar. 1982.

[4] ——, "Introduction to grey system theory," *Journal of Grey System*, vol. 1, pp. 1–24, Nov. 1989.

[5] [Online]. Available: http://androidideas.org/taskbomb/

[6] M. Sammarco, N. Belblidia, Y. Lopez, M. Dias de Amorim, L. H. M. K. Costa, and J. Leguay, "PePiT: Opportunistic Dissemination of Large Contents on Android Mobile Devices," in *ACM MobiOpp Demo Session*, Zurich, Switzerland, 2012.

[7] [Online]. Available: http://www.android-x86.org

[8] M. Sammarco, "Opportunistic Multi-Content Dissemination: Passive Monitoring and Adaptation to Network Conditions," Ph.D. dissertation, University Pierre and Marie Curie, May 2014.

[9] T. Claveirole and M. D. de Amorim, "Manipulating Wi-Fi packet traces with WiPal: design and experience," *Software Practice & Experience*, vol. 42, no. 5, pp. 585–599, May 2012.

[10] ——, "WiPal: IEEE 802.11 traces manipulation software," Jan. 2010. [Online]. Available: http://wipal.lip6.fr/

[11] A. Galati and C. Greenhalgh, "CRAWDAD data set nottingham/mall (v. 2013-02-05)," Downloaded from http://crawdad.cs.dartmouth.edu/nottingham/mall, Feb. 2013.

[12] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong, "CRAWDAD trace ncsu/mobilitymodels/gps/kaist (v. 2009-07-23)," Downloaded from http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels/GPS/KAIST, Jul. 2009.

[13] V. Borrel, F. Legendre, M. D. de Amorim, and S. Fdida, "Simps: using sociology for personal mobility." *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 831–842, 2009.

[14] J. Whitbeck, Y. Lopez, J. Leguay, V. Conan, and M. D. de Amorim, "Push-and-track: Saving infrastructure bandwidth through opportunistic forwarding," *Pervasive and Mobile Computing*, vol. 8, no. 5, pp. 682–697, Oct. 2012.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[16] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[17] F. Rebecchi, M. Dias de Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti, "Data offloading techniques in cellular networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, 2014.

[18] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, "Cellular traffic offloading through opportunistic communications: A case study," in *Proceedings of the 5th ACM Workshop on Challenged Networks*, ser. CHANTS '10, 2010.

[19] B. Han, P. Hui, V. S. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *IEEE Transactions on Mobile Computing*, vol. 11, no. 5, pp. 821–834, May 2012.

[20] Y. Li, M. Qian, D. Jin, P. Hui, Z. Wang, and S. Chen, "Multiple mobile data offloading through disruption tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 7, pp. 1–1, 2014.

[21] M. V. Barbera, A. C. Viana, M. D. de Amorim, and J. Stefa, "Data offloading in social mobile networks through VIP delegation," *Ad Hoc Networks*, pp. 92–110, 2014.

[22] Y. Chuang and K. C. Lin, "Cellular traffic offloading through community-based opportunistic dissemination," in *IEEE Wireless Communications and Networking Conference*, Paris, France, Apr. 2012.

[23] F. Rebecchi, M. D. de Amorim, and V. Conan, "Droid: Adapting to individual mobility pays off in mobile data offloading," in *IFIP Networking*, Trondheim, Norway, Jun. 2014.

[24] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Duke University, Tech. Rep., Jul. 2000.

[25] C. Boldrini and A. Passarella, "HCMM: Modeling spatial and temporal properties of human mobility driven by users' social relationships," *Computer Communications*, vol. 33, no. 9, pp. 1056–1074, Jun. 2010.

[26] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *OSDI*, 1999, pp. 173–186.

[27] V. Drabkin, R. Friedman, and M. Segal, "Efficient Byzantine broadcast in wireless ad-hoc networks," in *DSN*. IEEE Computer Society, 2005, pp. 160–169.

[28] "The Heartbleed Bug (http://heartbleed.com)."

[29] R. Lippmann, K. Ingols, C. Scott, and K. Piwowarski, "Validating and restoring defense in depth using attack graphs," *IEEE Military Communications Conference*, 2006.

[30] V. Bhandari and N. H. Vaidya, "On reliable broadcast in a radio network," in *PODC*, M. K. Aguilera and J. Aspnes, Eds. ACM, 2005, pp. 138–147.

[31] C.-Y. Koo, "Broadcast in radio networks tolerating Byzantine adversarial behavior," in *PODC*, S. Chaudhuri and S. Kutten, Eds. ACM, 2004, pp. 275–282.

[32] A. Pelc and D. Peleg, "Broadcasting with locally bounded Byzantine faults," *Inf. Process. Lett.*, vol. 93, no. 3, pp. 109–115, 2005.

[33] A. Maurer and S. Tixeuil, "Limiting Byzantine influence in multihop asynchronous networks," in *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS 2012)*, Jun. 2012, pp. 183–192.

[34] ——, "A scalable Byzantine grid." in *Proceedings of the 14th International Conference on Distributed Computing and Networking (ICDCN 2013)*, ser. Lecture Notes in Computer Science, vol. 7730. Springer, 2013, pp. 87–101.

[35] ——, "On Byzantine broadcast in loosely connected networks." in *Proceedings of the 26th International Symposium on Distributed Computing (DISC 2012)*, ser. Lecture Notes in Computer Science, vol. 7611. Springer, 2012, pp. 183–192.

[36] D. Dolev, "The Byzantine generals strike again," *Journal of Algorithms*, vol. 3, no. 1, pp. 14–30, 1982.

[37] M. Nesterenko and S. Tixeuil, "Discovering network topology in the presence of Byzantine faults," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 20, no. 12, pp. 1777–1789, December 2009. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPDS.2009.25

[38] H. Moniz, N. F. Neves, and M. Correia, "Turquois: Byzantine consensus in wireless ad hoc networks," *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010)*.

[39] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "ODSBR: An on-demand secure byzantine resilient routing protocol for wireless ad hoc networks," *ACM Transactions on Information and System Security*, vol. 11, pp. 18:1–18:35, 2007.

[40] T. Böhme, F. Göring, and J. Harant, "Menger's theorem," *Journal of Graph Theory*, vol. 37, no. 1, pp. 35–36, 2001.

[41] D. Kempe, J. Kleinberg, and A. Kumar, "Connectivity and inference problems for temporal networks," *Journal of Computer and System Sciences*, vol. 64, no. 4, pp. 820–842, 2002.

[42] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro, "Time-varying graphs and dynamic networks," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 27, no. 5, pp. 387–408, 2012.

[43] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on opportunistic forwarding algorithms," *TMC*, vol. 6, no. 6, pp. 606–620, 2007.

[44] M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols," *IEEE Trans. Softw. Eng.*, vol. 22, no. 1, pp. 6–15, Jan. 1996. [Online]. Available: http://dx.doi.org/10.1109/32.481513