Part-Whole Dissemination of Large Multimedia Contents in Opportunistic Networks

Nadjet Belblidia[a], Marcelo Dias de Amorim[a], Luís Henrique M. K. Costa[b], JérémieLeguay[c], Vania Conan[c]

[a] *UPMC Sorbonne Universités, 4 Place Jussieu, 75005 Paris, France.* {*nadjet.belblidia, marcelo.amorim*}*@lip6.fr*
[b] *COPPE/PEE, Universidade Federal do Rio de Janeiro, P.O. Box 68504, 21945-970 Rio de Janeiro, Brasil. luish@gta.ufrj.br*
[c] *Thalès Communications, 160, Boulevard de Valmy, 92700 Colombes, France.* {*jeremie.leguay, vania.conan*}*@fr.thalesgroup.com*

Abstract

A common assumption in intermittently-connected (or opportunistic) mobile networks is that any contact has enough capacity to transfer the required amount of data. Although such an assumption is reasonable for analytical purposes and when contents are small, it does not hold anymore when users produce contents that are larger than the capacity of a contact. In such a case, users must slice data and send fragments separately, which allows better use of short contacts and progressive dissemination of large contents data pieces. The main question here is to design the best strategy for deciding which piece(s) to transmit whenever nodes meet. In addition, although small pieces imply a better use of short contacts, they generate more overhead due to the headers required at each piece. In this paper, we investigate these two issues: piece size selection and piece selection strategy. First, we theoretically define the global goodput of the system that defines the tradeoff between the size of the shortest contact that can be considered as useful and piece overhead. Results from real-world traces show that, for reasonable header size, the piece size can be selected out of a large range of values without significantly impacting the results. Second, we present the design and evaluation of PACS (Prevalence-Aware Content Spreading), a completely distributed algorithm that selects pieces to transfer based on their popularity. We evaluate the performance of PACS using both synthetic and real traces from intermittently-connected networks. When compared with sequential and randomized solutions, we show that PACS significantly outperforms these approaches both in terms of latency to achieve full dissemination and ratio of effective contacts. Moreover, PACS achieves performance levels that are extremely close to a centralized oracle-based solution.

Key words: Intermittently-connected mobile networks, multimedia content dissemination, peer-to-peer.

1. Introduction

Important advances in the area of opportunistic networks have been achieved including the conception of applications to enable content sharing among users on the move [1, 2, 3]. In our daily lives, users generate, consume, and share contents that are becoming increasingly larger. We address the following question: *how to efficiently disseminate such large contents in opportunistic networks when contacts have limited capacity?* This is a realistic situation, as portable devices such as smartphones and compact cameras are now able to generate high-definition videos that are resource-consuming. As an idea, average standard videos on YouTube are 10MB long [4]; in HD quality, this value goes up to 40MB. If we consider Bluetooth as the underlying transport technology (as suggested in several proposals), transferring such amounts of data opportunistically would require contacts of 80 to 320 seconds, at best.

A few experimental initiatives have shown that most contact durations in human-driven opportunistic networks fall under the minute [5, 6, 7].[1] For example, Gaito et al. show in their experiment that more than 50% of the contacts last for less than 1 minute (they found a median contact time of 48 seconds). Trying to transfer large contents during these short-lived encounters becomes impractical, as two main limitations rise. First, nodes that experience short contacts frequently might never receive the data. Second, transfer opportunities are wasted leading to poor overall performance. To optimize data dissemination in such scenarios, it is fundamental to adapt the amount of transmitted data to the contact capacity. Hence, nodes must slice the data and send fragments separately. The main challenge when disseminating fragmented data is to decide which piece(s) should be sent when two nodes meet. [2]

Before addressing the piece selection problem, the first essential point to investigate is *how to determine the fragment (piece) size*. One solution is to specifically adapt the piece size to each contact capacity. In other terms, nodes must take into account the capacity of each contact when sending any content. If the content is small enough to be transmitted during the contact, the content is fully sent. Otherwise, the content is divided into several pieces so that at least one piece can be transmitted during the contact. This solution is not straightforward since the contact capacity characterization needs to be very accurate. Therefore, it highly depends on the underlying technology. Another solution is to have a standard piece size; when nodes generate large contents, they automatically divide the contents into

[1] Other fundamental papers could not show such a behavior as they relied on beaconing periods of 120 seconds or more [8, 9].

[2] This paper is a significant extension of our previous paper "PACS: Chopping and shuffling large contents for faster opportunistic dissemination", published at *IFIP WONS* 2011 [10].

pieces of equal size. In this paper, we consider this second solution to determine the standard piece size. In Section 4, we formally bring out the tradeoff to deal with between turning small contacts into useful and increasing the payload. We have used movement traces obtained from RollerNet to study the impact of piece size selection on a real world dissemination scenario. RollerNet trace was collected from an intermittently-connected mobile network formed between 62 people during a rollerblading tour in the streets of Paris, which lasts for 3 hours [5]. Interestingly, results show that as header remains reasonable, the piece size can be selected from a large range of values without significantly impacting the results.

To address the problem of *which piece(s) should be sent when two nodes meet*, one possibility is to rely on a naive approach and transfer pieces in a sequential order, i.e., nodes disseminate the pieces with the lowest identifiers first (see Section 5). As we will show later, the main problem with this approach is that it does not capture the conditions of the network and leads to poor dissemination ratio. Another possibility is to disseminate pieces in a uniformly-distributed random way, but it does not capture contact patterns either. In this paper, we show that: (i) the order of piece dissemination matters, (ii) bad piece selection can lead to ineffective contacts, and (iii) uniform random selection is not enough. To our knowledge, no previous work has addressed this problem.

In order to counterpart the abovementioned issues, we propose PACS (Prevalence-Aware Content Spreading), a popularity-based strategy to select pieces to be exchanged between neighbors solely based on node-local information. Through their successive contacts, nodes keep track of the dissemination level of the pieces throughout the network and use this information to transfer less prevalent pieces first. To this end, nodes exchange a small boolean vector when in contact. By combining such vectors over time, nodes are able to build a popularity map of pieces in the network. We show that such a simple local strategy significantly increases the system performance. We evaluate PACS using both synthetic and real-world mobility traces from intermittently-connected networks. Synthetic user movements are generated using the random trip model [11] and the community-based mobility model proposed in [12]. Additionally, we have also used the RollerNet trace described above.

In summary, the key contributions of PACS are:

- **Higher heterogeneity of pieces in the network.** PACS prevents nodes form getting the same pieces first, which leads to quick increase in the number of infected nodes.
- **More useful contacts.** PACS leads to much higher contact effectiveness, i.e., it reduces the number of contacts that cannot be used because nodes have the same pieces.
- **Reduced dissemination delay.** By turning more contacts into effective opportunities, PACS significantly reduces the latency for the contents to be fully pushed to all nodes.

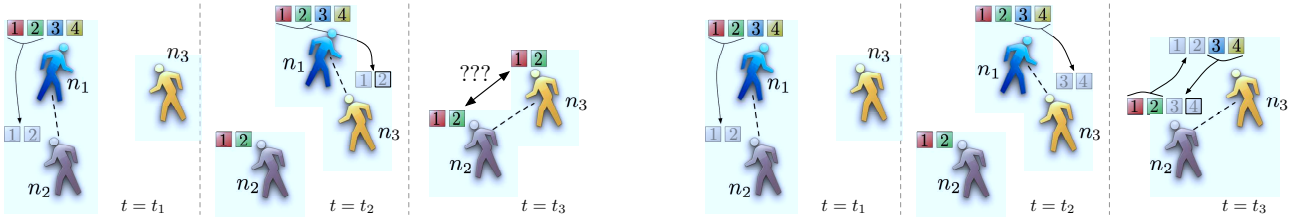
The remainder of the paper is structured as follows. In Section 2, we give an overview of related work. We briefly describe the problem of content dissemination in opportunistic networks and present our network model in Section 3. We formalize the problem of piece dimensioning in Section 4 and describe the basic piece dissemination strategies, namely sequential and random strategies, in Section 5. We present PACS in Section 6 and evaluate its features in Sections 7, 8, and 9. Finally, in Section 10, we conclude the paper and raise future research directions.

2. Related work

Data broadcasting in opportunistic and ad hoc networks has been the subject of several works. The proposed approaches can be classified in four main categories: simple flooding, probability based, area based, and neighbor knowledge [13]. In addition, a new data dissemination category based on network coding emerged recently [14, 15]. The main objective of all these solutions is to achieve an efficient dissemination while minimizing the number of transmissions in the network. This is done by selecting the best relay nodes among all the neighbors an infected node has. Nevertheless, all these approaches assume that any contact is long enough to transfer the data under consideration. This problem is somehow complementary to the one addressed in our paper. Indeed, these solutions answer to the question of how to select relay nodes while we address the question of how to select the piece to transfer once the relay node is already selected.

Pitkänen et al. studied the impact of data fragmentation in one-to-one opportunistic network communications [16]. They considered two fragmentation strategies: reactive fragmentation and proactive fragmentation. In reactive fragmentation, the sender starts transmitting the data until it is interrupted by the link failure caused by the end of the contact. In proactive fragmentation, the source node divides the data into pieces of standard size (based on the expected average contact capacity). They concluded that the reactive fragmentation with predefined fragment boundaries allows significant improvements in one-to-one communications. In this paper, we show that even simple proactive fragmentation can improve one-to-all communications (data dissemination in our case). We found that a large range of piece sizes allows reducing the overall dissemination delay (for more details, please refer to Section 9.2).

As discussed in Section 6, PACS, our piece selection proposal, is inspired by BitTorrent. Several solutions have been proposed to adapt BitTorrent to opportunistic and ad hoc networks [17, 18, 19]. Most of these adaptations, however, aim at constructing and maintaining an overlay network that enables multi-hop message routing. In other terms, nodes do not need to be direct neighbors to become peers. Our solution, in turn, uses the network layer and the immediate communication capabilities of the nodes to disseminate data. Nadan et al. proposed SPAWN, a cooperative strategy for content downloading in vehicular networks [20]. The piece selection scheme used in SPAWN is based on a proximity-driven strategy called rarest-closest. Such a strategy selects the rarest pieces and then ranks them based on the distance to the closest peer that has that piece. This solution shares with PACS the same motivations, i.e., they



(a) Sequential. (b) Random.

Figure 1: A motivating example. Selecting the pieces to transfer is fundamental to efficient dissemination of fragmented contents.

prioritize rarer pieces and consider peer location. SPAWN and PACS differ however on a fundamental aspect. SPAWN, as the abovementioned solutions, constructs an application-layer overlay that does not limit the peer selection to the one-hop neighborhood. Hence, it needs an underlay routing protocol that maintains multi-hop routes between peers.

Some other solutions implemented file swarming by only considering one-hop communications [21, 22]. Both solutions use uniformly-distributed random piece selection. Nevertheless, they use network coding in order to mitigate the coupon collection problem by increasing piece heterogeneity. Finally, some papers presented different architectures to enable mobile peer-to-peer distribution of large contents [23, 24]. In both architectures, contents are exchanged opportunistically when nodes are within communication range. However, the piece selection strategy differs. Jung and al. used the random selection strategy [23] whereas Helgason et al. presented an implementation of the sequential strategy using a pull-based architecture [24]. We show in this paper that a more sophisticated piece selection strategy can enhance such architectures.

3. Content Spreading in Opportunistic Networks

In this section, we provide all the necessary background before introducing the piece dimensioning problem and the dissemination algorithms. 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. In this context, we address two problems. First, having a clue on the contact capacities, *how to efficiently select the piece size*? Second, given the pieces and a contact opportunity, *which subset of these pieces should be transferred if the contact is not sufficient to transmit them all*?

3.1. 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 paper.

We show in Fig. 1(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.

The ideal case would have been the one in Fig. 1(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. As we will show later in this paper, PACS is a generalization to the solution shown in Fig. 1(b).

3.2. Network model and assumptions

Let $\mathbf{N} = \{n_0, n_1, \ldots, n_N\}$ be the set of N nodes in the network. Nodes are mobile, but we do not assume any a priori knowledge of mobility patterns. For the sake of simplicity, we assume that all nodes in the network are interested in the unique content $\mathbf{C}$ that is initially only available at a single node. Without loss of generality, we call this node the data source and denote it as n_0. The generalization to any number of data sources and contents is straightforward.

The data source chops the content into K pieces of equal size. The number of pieces is deduced after selecting piece size (See Section 4, for more information about how to accurately determine the piece size). Pieces are sequentially identified as $\mathbf{C} = \{c_0, c_1, \ldots, c_{K-1}\}$. 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 stores locally an *availability bitmap vector* $\mathbf{a}_{n_i} = \{a_0, \ldots, a_{K-1}\}$, where $a_k = 1$ if the node has piece c_k, and $a_k = 0$ otherwise. The necessary contact time to transfer one piece is noted τ. We call this a contact slot. Thus, a contact duration t can be used to transfer $\lfloor \frac{t}{\tau} \rfloor$ pieces.

All the variables are summarized in Table 1.

Table 1: Summary of the variables.

Variable	*Definition*
N	Set of nodes in the network
N	Number of nodes in **N**
n_0	Data source
C	Content to be disseminated
K	Number of pieces that compose **C**
c_i	ith piece of **C**
τ	Contact slot (time to transfer one piece)
$\mathbf{a}_{n_j}$	Availability bitmap of node n_j

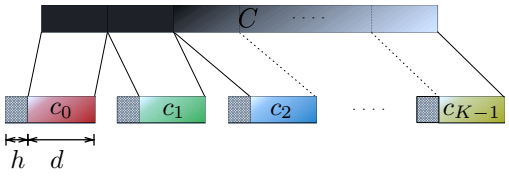


Figure 2: Chopping a large content into several pieces.

4. Dimensioning content pieces

The first step to deal with when considering the dissemination of large contents in opportunistic networks is the the choice of the piece size. Content C is chopped into K pieces of the same size (c.f., Fig. 2). In addition to a data block of size d, each piece includes an overhead of fixed size h. Besides the source ID, the overhead could include useful information to line up the pieces into their correct positions when rebuilding contents as the content ID and the sequence number of the piece.

On one the hand, the size of the piece must be small to take advantage from most of contact opportunities. Indeed, the smaller the piece, the larger the percentage of contacts able to transmit it. Nevertheless, when the piece is too small, the goodput decreases since the overhead increases. On the other hand, if the piece is large, the overhead introduced remains negligible – but the number of useful contacts decreases. Therefore, there is a tradeoff to deal with when selecting the piece size.

As illustrated in Fig. 2, C is chopped into K pieces $\{c_0, c_1, \ldots, c_{K-1}\}$ of size p. Each piece contains a data block of size d and a header of size h. Hence, $p = h + d$. The distribution of contact capacity is represented by the complementary cumulative distribution function F shown in Fig. 3. For pieces of size p, the proportion of useful contacts (those that are enough to transfer a piece of size p) is $F(p)$. The tradeoff can be expressed as follows. As the piece size tends to the minimum contact capacity (moving from the right to the left on the x-axis), more contacts are used ($F(p)$ increases) but more overhead is introduced ($\frac{d}{h+d}$ decreases). As the piece size tends to the maximum contact capacity (moving from the left to the right on the x-axis), fewer contacts are used ($F(p)$ decreases) but less overhead is introduced ($\frac{d}{h+d}$ increases).

Assuming that there is always at least one piece to exchange each time a contact happens between two nodes (this gives us an upper bound), the global goodput $\mathcal{G}$ can be expressed as follow:

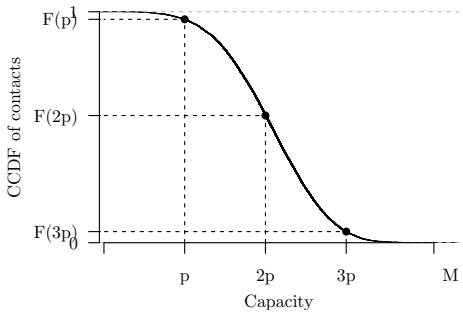


Figure 3: Example of contact capacity CCDF. Dimensioning problem parameters. p is the piece size. $p = h + d$.

$$\mathcal{G} = \frac{d \times \mathcal{S}}{T}, \tag{1}$$

where T is the total contact time and $\mathcal{S}$ is the number of useful contacts able to transmit one piece of size $p = h + d$.

If all the contacts had the same capacity p, $\mathcal{S}$ would be equal to $m \times F(p)$ (m being the total number of contacts). However, a contact of more than $2p$ would be able to transmit 2 pieces. This represents $F(2p)$ of contacts. A contact of more than $3p$ capacity could transmit 3 pieces ($F(3p)$ of contacts), and so on. Hence, $\mathcal{S}$ can be writen as:

$$\mathcal{S} = m \times \sum_{i=1}^{\lfloor \frac{M}{p} \rfloor} F(i \times p), \tag{2}$$

where M is the maximum contact capacity (c.f., Fig. 3). When we replace $\mathcal{S}$ in eq. 1, we get:

$$\mathcal{G} = d \times m \times \frac{1}{T} \times \sum_{i=1}^{\lfloor \frac{M}{h+d} \rfloor} F(i(h+d)). \tag{3}$$

T and m being fixed, we must find the maximum value of the following function g to maximize the global goodput $\mathcal{G}$:

$$g(x) = x \times \sum_{i=1}^{\lfloor \frac{M}{h+x} \rfloor} F(i(h+x)). \tag{4}$$

5. Basic content dissemination strategies

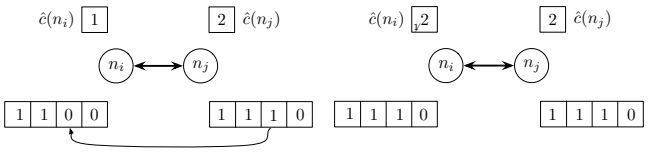
We now detail the operation of the basic piece selection strategies. A piece selection strategy specifies the piece to transfer during a contact slot. We call "basic" strategies the sequential one illustrated in Section 3.1 and a randomized one where pieces to be transferred are selected following a uniform law.

5.1. Sequential content dissemination

In the sequential strategy, nodes transfer pieces to neighbors in an increasing order of identifiers. This implies that if a node has piece c_j, it necessarily has pieces c_k, $\forall 0 \leq k \leq j$. We note $\hat{c}_{n_i}$ as the largest identifier of pieces owned by node n_i,

Algorithm 1 n_i sequential strategy

```
1: while contact_with(n_j) do
2:    receive_from(n_j, ĉ(n_j));
3:    if (ĉ(n_j) < ĉ(n_i)) and (initiate_connexion_with(n_j)) then
4:       c_{s_{i→j}} ← c_{ĉ(n_j)+1};
5:       send_to(n_j, c_{s_{i→j}});
6:    else
7:       if (ĉ(n_j) > ĉ(n_i)) and (connexion_initiated_by(n_j)) then
8:          receive_from(n_j, c_{s_{j→i}});
9:          if (c_{s_{j→i}} = c_{ĉ(n_i)+1}) then
10:            ĉ(n_i) ← ĉ(n_i) + 1;
11:         else
12:            ignore(c_{s_{j→i}});
13:         end if
14:      end if
15:   end if
16: end while
```



(a) State 1: selecting pieces. (b) State 2: updating local variables.

Figure 4: Piece selection using the sequential strategy. Initially, n_i has pieces $\{c_0, c_1\}$ ($\hat{c}_{n_i}$=1). n_j has pieces $\{c_0, c_1, c2\}$ ($\hat{c}_{n_j}$= 2).

i.e., $\hat{c}_{n_i} = j$ if $a_k = 1, \forall k \leq j$ and $a_k = 0, \forall k > j$. Initially, all nodes in the network are looking for the first piece (i.e., c_0) except the data source n_0 that already has all pieces. Formally, $\hat{c}_{n_i} = -1, \forall n_i \in \mathbf{N} \backslash n_0$ and $\hat{c}_{n_0} = K - 1$.

When two nodes n_i and n_j meet, they exchange their corresponding $\hat{c}$. Consider first the case where $\hat{c}_{n_i} > \hat{c}_{n_j}$, which means that n_i has at least one piece that n_j does not have. As long as the contact duration allows, node n_i transfers pieces following the sequence $c_{\hat{c}_{n_j}+1}, c_{\hat{c}_{n_j}+2}, \ldots, c_{\hat{c}_{n_i}}$. If $\hat{c}_{n_i} < \hat{c}_{n_j}$, the same is done but from n_j to n_i. At each transfer, the receiving node increments its corresponding $\hat{c}$. Note that if $\hat{c}_{n_i} = \hat{c}_{n_j}$, the contact becomes useless as the nodes have exactly the same contents. For a contact of duration t, the maximum number of pieces transferred is $\min\{|\hat{c}_i - \hat{c}_j|; \lfloor t/\tau \rfloor\}$.

This strategy is illustrated in Fig. 4. The content is composed of four pieces ($K = 4$). In this example, the only possible exchange is transferring the piece c_2 from n_j to n_i. Algorithm 1 details the strategy.

5.2. Uniform random content dissemination

The idea behind the uniform content spreading strategy is to select, among the pieces a neighbor has not received yet, the ones to be transferred in a uniformly-distributed random way. When nodes n_i and n_j meet, they exchange their availability vectors $\mathbf{a}_{n_i}$ and $\mathbf{a}_{n_j}$ (as defined in Section 3.2). Node n_i (resp. n_j) computes $\mathbf{a}_{n_i} \wedge (\neg \mathbf{a}_{n_j})$ (resp. $\mathbf{a}_{n_j} \wedge (\neg \mathbf{a}_{n_i})$), which gives the candidate pieces to be transferred ($\wedge$ stands for the "and" operator and $\neg$ is "not"). During the contact, one or more of these candidate pieces are chosen to be transferred based on a uniformly-distributed random way. After one round of exchanges, nodes update their availability vectors as:

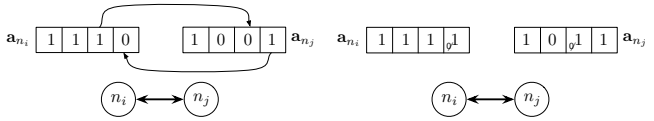
$$\begin{aligned} \mathbf{a}_{n_i} &\leftarrow \mathbf{a}_{n_i} \vee \mathbf{i}_{c_{j\rightarrow i}}, \\ \mathbf{a}_{n_j} &\leftarrow \mathbf{a}_{n_j} \vee \mathbf{i}_{c_{i\rightarrow j}}. \end{aligned} \tag{5}$$

where $\mathbf{i}_{c_{i\rightarrow j}}$ and $\mathbf{i}_{c_{j\rightarrow i}}$ are vectors of K elements with all positions equal to 0 except the position relative to the piece just received, which is set to 1 ($\vee$ stands for the "or" operator).

We illustrate the algorithm in Fig. 5. Node n_i (resp. n_j) has pieces $\{c_0, c_1, c_2\}$ (resp. $\{c_0, c_3\}$) as shown in the availability vectors (Fig. 5(a)). After exchanging their vectors, the only piece n_j could send to n_i is c_3 when n_i could randomly select one of the pieces $\{c_1, c_2\}$ to send it to n_j. Assume that the contact lasts for two slots. Hence, two pieces can be exchanged. Suppose that n_j sends c_3 to n_i during the first slot and that n_i sends piece c_2 during the second slots. After piece transfers, each node updates its vector. The availability vectors become $\mathbf{a}_{n_i} = \{1, 1, 1, 1\}$ and $\mathbf{a}_{n_j} = \{1, 0, 1, 1\}$. The strategy is fully detailed in Algorithm 2.

Algorithm 2 n_i Uniform random strategy

```
1: while contact_with(n_j) do
2:    receive_from(n_j, a_j);
3:    if (a_i ∧ (¬a_j) ≠ ∅) and (initiate_connexion_with(n_j)) then
4:       c_{s_{i→j}} ← radom_selection_from(a_i ∧ (¬a_j));
5:       send_to(n_j, c_{s_{i→j}});
6:    end if
7:    if (a_j ∧ (¬a_i) ≠ ∅) and (connexion_initiated_by(n_j)) then
8:       receive_from(n_j, c_{s_{j→i}});
9:       i_{j→i} ← {i_0, ..., i_{K−1}}; i_k = 0, ∀k < K (k ≠ s_{j→i}) and i_{s_{j→i}} = 1
10:      a_i ← a_i ∨ i_{j→i};
11:   end if
12: end while
```



(a) State 1: selecting pieces. (b) State 2: updating local vectors.

Figure 5: Piece selection using the uniform random strategy. Initially, n_i has pieces $\{c_0, c_1, c_2\}$ and n_j has pieces $\{c_0, c_3\}$.

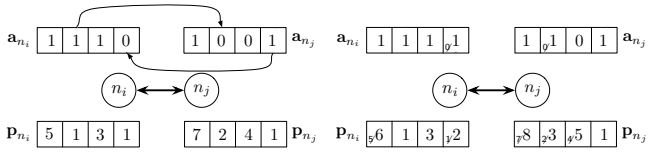
6. PACS: Prevalence-Aware Content Spreading

The goals of PACS are to achieve fast content dissemination while keeping the overhead low and making better use of contact opportunities. The challenges of conceiving such a system are mainly twofold. First, nodes must have a clue on the dissemination progress of each piece, so that they can appropriately prioritize their transmissions. Second, the dissemination information must remain local to reduce the overhead and achieve a scalable solution.

In PACS, in addition to the availability vector, node n_i also keeps a prevalence vector $\mathbf{p}_{n_i} = \{p_0, p_1, \ldots, p_{K-1}\}$. As it will become clearer later, the goal of $\mathbf{p}_{n_i}$ is to give a local view of

Algorithm 3 n_i PACS strategy

```
1: while contact_with(n_j) do
2:    receive_from(n_j, a_{n_j});
3:    p_{n_i} ← p_{n_i} + a_{n_j};
4:    if (a_{n_i} ∧ (¬a_{n_j}) ≠ ∅) and (initiate_connexion_with(n_j)) then
5:       c_{s_{i→j}} ← prevalence_selection_from((a_{n_i} ∧ (¬a_{n_j})), p_{n_i});
6:       send_to(n_j, c_{s_{i→j}});
7:    end if
8:    if (a_{n_j} ∧ (¬a_{n_i}) ≠ ∅) and (connexion_initiated_by(n_j)) then
9:       receive_from(n_j, c_{s_{j→i}});
10:      i_{c_{j→i}} ← {i_0, ..., i_{K−1}}; i_k = 0, ∀k < K (k ≠ s_{j→i}), i_{s_{j→i}} = 1
11:      a_{n_i} ← a_{n_i} ∨ i_{c_{j→i}};
12:   end if
13: end while
```



(a) State 1: select pieces. (b) State 2: update local vectors.

Figure 6: Piece selection using PACS. Initially, n_i has pieces $\{c_0, c_1, c_2\}$ and n_j has pieces $\{c_0, c_3\}$.

the prevalent pieces in the network. Initially, all nodes have an empty prevalence vector. When nodes n_i and n_j meet, they exchange their availability vectors, exactly in the same way as the uniform content dissemination strategy. They also update their prevalence vectors respectively as:

$$\begin{aligned} \mathbf{p}_{n_i} &\leftarrow \mathbf{p}_{n_i} + \mathbf{a}_{n_j}, \\ \mathbf{p}_{n_j} &\leftarrow \mathbf{p}_{n_j} + \mathbf{a}_{n_i}. \end{aligned} \tag{6}$$

Among the candidate pieces to be transferred, nodes select the one with the lowest prevalence. In case of tie, a piece is chosen in a uniformly distributed random way. Let $c_{s_{i\to j}}$ be the piece sent by n_i to n_j and $c_{s_{j\to i}}$ be the piece sent by n_j to n_i. Once this step done, nodes update their availability vectors as indicated in Equation 5.

In the very beginning, the prevalence vector has a limited influence on the selection algorithm but gains importance as nodes move and exchange pieces. We show an example in Fig. 6. After exchanging their availability vectors, nodes update their prevalence vectors as indicated in Equation 6 ($\mathbf{p}_{n_i} = \{6, 1, 3, 2\}$ and $\mathbf{p}_{n_j} = \{8, 3, 5, 1\}$). Similarly to the previous examples, we assume that contact last for 2 slots. Then, n_j transfers to n_i the piece c_3 that is the only piece it is able to select, while n_i chooses the less prevalent piece from $\{c_1, c_2\}$ to send to n_j. According to $\mathbf{p}_{n_i}$, piece c_1 is less prevalent than piece c_2. Node n_i sends c_1 to n_j. Once the exchanges are done, the respective availability vectors are set to $\mathbf{a}_{n_i} = \{1, 1, 1, 1\}$ and $\mathbf{a}_{n_j} = \{1, 1, 0, 1\}$. The strategy is described in Algorithm 3.

Note that PACS has some similarities with peer-to-peer systems, notably BitTorrent [25, 26, 27]. Indeed, PACS uses a BitTorrent-like content swarming where data is divided into several pieces. When two nodes are in range of each other, they try to exchange the pieces with the lowest prevalence first. This corresponds somehow to the rarest-first algorithm used in BitTorrent. Nevertheless, the notion of rarest piece is essentially different in the two cases. In BitTorrent, each peer maintains a list of the number of copies in its peer set. This list corresponds to the prevalence vector described in PACS but contains exactly the number of copies in the peer set (neighborhood). In PACS, instead, nodes update their prevalence vector each time they initiate a connection with another node. Even if both strategies give the node an egocentric view of the rarest pieces, PACS adapts the algorithm to counterbalance the instability of a node's neighborhood due to the dynamics of the environment. Indeed, the nodes that are the most represented in the prevalence vector are those encountered often and/or during longer time intervals.

7. Evaluation framework

In this section, we summarize the simulation and model parameters. We use the ONE [28] simulator with both mobility models and real movement trace based simulations.

7.1. Simulation parameters

We study the impact of the following main parameters:

Area size. We consider two scenarios with the following area sizes: 300m× 300m and 1,000m×1,000m. The first area is of the size of a train station when the second area is large as a downtown area.

Number of nodes. The number of nodes varies between 100 and 2,500. By default, the number of nodes is set to 250. This parameter, associated to the area size, determines both the network density and the network diameter.

Number of data sources. By default, we consider a unique content originally available at a single data source. When studying the impact of the number of initial copies, we vary the number of data source between 1 and 250 nodes.

Number of piece sources. The content pieces are generated at a single data source by default. When we evaluate the impact of the initial piece dispersion on the dissemination delay, we vary the source of the different pieces from all the pieces generated at a single source (number of piece source set to 1) to each piece generated at a different source (number of piece source set to the number of pieces).

Content size. The content size is set to either 12MB or 48MB. We consider these values to fit a realistic scenario of video dissemination. As observed in [4], videos in YouTube have a mean duration of 4.15 minutes for an average size of 10MB. In our simulations, a 12MB-file represents a standard definition video, while a 48MB-file is a high-definition video.

Piece size. We investigate the impact of the piece size on the effectiveness of the algorithms. The piece size is incremented exponentially from 3kB to 3MB. By default, the piece size is set to 384kB. The piece size together with the content size determines the number of pieces.

Header size. When we evaluate the impact of the piece size, we consider two header size values: 56B and 512B. By default, we consider that pieces have no header.

Table 2: Simulation parameters.

Factors	Area size	300m×300m, 1,000m×1,000m
	Number of nodes	100, **250**, 500, 1,000, 2,500
	Number of data sources	**1**, 2, 5, 10, 25, 50, 100, 150, 200, 250
	Number of piece sources	**1**, 2, 4, 8, 16, 32
	Data size	12MB, 48MB
	Piece size	3kB, 6kB, 12kB, 48kB, 96kB, 192kB, **384kB**, 768kB, 1.5MB, 3MB
	Header size	**0B**, 56B, 256B
Parameters of the models	Range	10m
	Moving speed	[0.5, 1.5] m/s
	Throughput	125 kBps
RollerNet configuration	Number of nodes	62
	Trace duration	3 hours
	Throughput	125 kBps

These parameters are summarized in Table 2, where bold values stand for the default ones.

7.2. Parameters of the mobility models

We used two mobility models for the simulations. First, nodes follow the random trip model. We only consider the steady state of the random waypoint by applying the formulas described in [11]. The second model is the community-based model formulated by Musolesi et al. [12].

For both models, nodes move at walking speed (between 0.5m/s and 1.5m/s). Two nodes are able to communicate when within communication range of 10m. Data is transferred at a throughput of 125KBps. In addition, each model has specific parameters. For random trip, nodes may pause between two trips. Node pause time is uniformly picked in the interval $[0, 120]$s. In the community-based model, nodes are grouped together based on social relationship among individuals. The initial number of groups is set to 50. Groups are mapped onto a topographical space corresponding to cells. The number of cells in the area is set to 3×3. Table 2 summarizes the parameters of the models.

7.3. Real-world trace configuration

We use the RollerNet trace to evaluate the performance of the spreading strategies in real-world environment [5]. The trace has been generated through contact logs between Intel iMote nodes (equipped with a Bluetooth interface). Each iMote performs regular scans and registers the MAC addresses of the responding devices around. The RollerNet trace has been collected during a rollerblading tour in Paris. iMotes were distributed to 62 participants and the total duration of the tour was about three hours. This trace is publicly available to the community through the Crawdad repository.[3]

The number of nodes is set to the number of participants in the experiment (i.e., 62). The transmission throughput of nodes is set to 125KBps that correspond to the nominal Bluetooth throughput. At each simulation run, we pick a different node to play the role of the data source. The trace configuration is also summarized in Table 2.

7.4. Benchmarking

We compare PACS with both the sequential and the uniform random strategies as described in Section 5. Besides these strategies, we also consider a centralized strategy where a central entity maintains a global prevalence vector. The global prevalence vector is used to select the piece to be transferred by nodes in the same way as in PACS. Nevertheless, it is only updated when a node receives a piece. The global prevalence vector reflects exactly the current dissemination state of each piece in the network. We call this strategy the *Oracle*. Obviously, deploying such a centralized strategy is impracticable in a real opportunistic network. We use it for comparison purposes only.

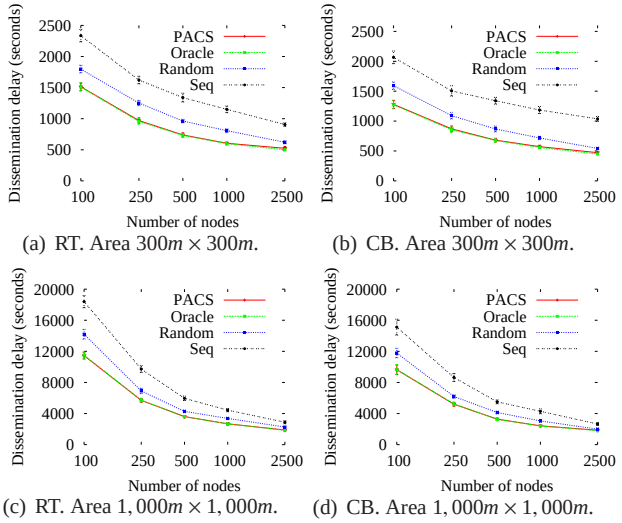
8. Synthetic mobility evaluation

We use two mobility models to generate synthetic traces. First, we study the simple case of mobility induced by the random trip model. Second, we consider the community-based mobility model, a more elaborated model founded on social network theory. Plots represent average results upon 20 different runs. Parameters are detailled in Section 7.2.

8.1. Impact of network density and diameter

We vary both the area size and the number of nodes to study the dissemination delay of a 12MB-file (Fig. 7). We define the dissemination delay as the required duration for the content to be received by all the nodes in the network. It is the elapsed time between the transmission of the first piece to the first node and the reception of the last piece by the last node. We also measure the contact effectiveness (Fig. 8). The contact effectiveness is the ratio of the time used for transfers over the total contact durations (in the period comprised between the first and the very last piece transfers). It indirectly measures the availability of new pieces when nodes meet. An effectiveness closer to zero means that nodes meet but seldom have pieces to transfer, while effectiveness closer to one reflects frequent exchanges. As expected, for the four strategies, the larger the number of nodes (denser network), the smaller the dissemination delay and contact effectiveness. This is due to the increase of the number of contact opportunities in denser networks. The sequential strategy leads to the worst performance. Even if the difference between the strategies is accentuated in sparse zones (with fewer nodes in the network), we can observe the same

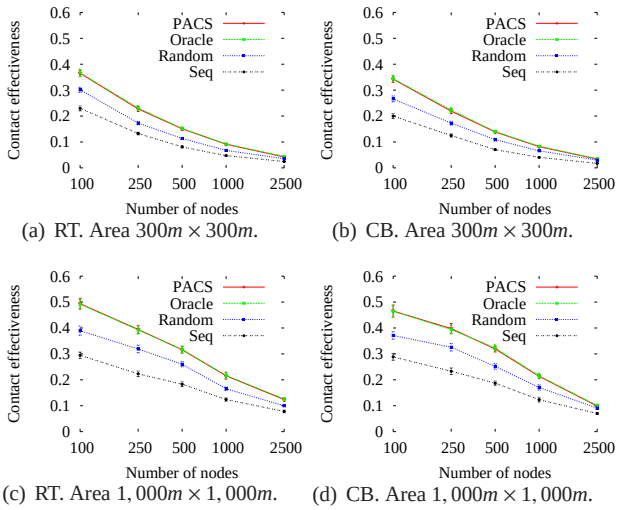
[3] http://crawdad.cs.dartmouth.edu/meta.php?name=upmc/rollernet



(a) RT. Area $300m \times 300m$. (b) CB. Area $300m \times 300m$.

(c) RT. Area $1,000m \times 1,000m$. (d) CB. Area $1,000m \times 1,000m$.

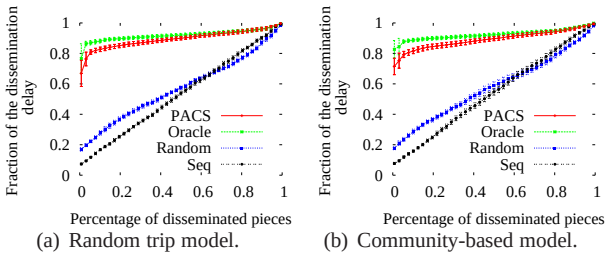
Figure 7: Dissemination delay according to the number of nodes. Dissemination of a 12MB data divided on 32 pieces of 384KB. Random trip (RT) versus community-based model (CB).



(a) RT. Area $300m \times 300m$. (b) CB. Area $300m \times 300m$.

(c) RT. Area $1,000m \times 1,000m$. (d) CB. Area $1,000m \times 1,000m$.

Figure 8: Contact effectiveness according to the number of nodes. Dissemination of a 12MB data divided on 32 pieces of 384kB. Random trip (RT) versus community-based model (CB).



(a) Random trip model. (b) Community-based model.

Figure 9: Piece dissemination evolution. 250 nodes. Dissemination of a 48MB data divided on 128 pieces of 384kB. Area $1,000m \times 1,000m$.

tendency in dense networks (Fig. 7(a) and Fig. 7(b)). Regardless of the number of nodes and the area size, PACS performs better than the sequential and the random strategies, reducing by about a half in average the dissemination delays. Furthermore, the results of PACS tend to the ones obtained using the Oracle strategy.

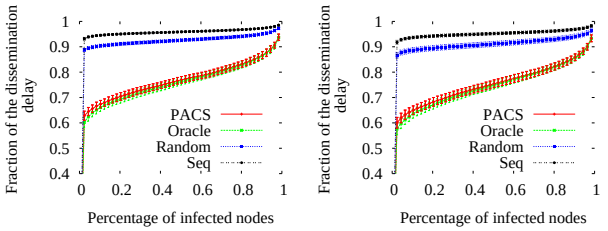
8.2. Impact of the strategy on the evolution of piece dissemination

In order to understand the reason of such a difference in the dissemination delay between the strategies, we first compare the strategies regarding the piece dissemination evolution. Fig. 9 shows the proportion of time required, out of the total time, to fully disseminate a given percentage of pieces. The total time corresponds to the dissemination delay. The piece dissemination is faster with the random and sequential strategies. Indeed, all the nodes get the first piece after 17% of the total time for the random and only after 7% of the total time for the sequential. This reflects the fact that all nodes start by getting the same pieces with those strategies. Conversely, with PACS and Oracle, nodes start by getting different pieces and no pieces are fully disseminated before 82% of the total time for Oracle and 71% of the total time for PACS.

But what matters is the global behavior of the dissemination evolution. Fig. 10 shows the proportion of time required, among the total time, to infect a definite percentage of nodes. A node is infected when it gets all the pieces. Regardless the mobility model, we observe two different behaviors. Clearly, with PACS and Oracle, nodes are infected very quickly compared to the random and sequential strategies. With PACS and Oracle, the first node is infected around half of the total time. On the other hand, this first node is only infected at 80% of the total time with the random and sequential strategies. Moreover, when the simulation achieves 90% of the total time, only 1.6% (resp. 29%) of nodes are infected with the sequential strategy (resp. random strategy), whereas 96% of nodes are already infected with PACS and Oracle. This result indicates that PACS could be more robust to a premature departure of the source node from the network since other nodes are able to play the role of seeds earlier in the dissemination process.

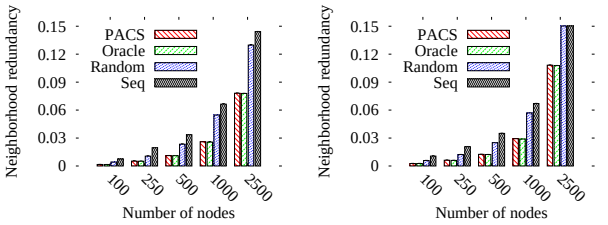
8.3. Impact of the strategy on the neighborhood redundancy

We define the neighbor redundancy as the average fraction of useless contacts. A contact is considered useless if the two nodes involved in it have no pieces to exchange. We consider the dissemination of a 48MB-file divided in 128 pieces. Fig 11 shows the neighborhood redundancy according to the number of nodes in the network. For all strategies, the nodes face more useless connections when the network is denser. Indeed, with the random trip model for example (Fig. 11(a)), only 1% or less of the contacts are useless when we have 100 nodes in the network. This proportion is 10 times larger for 250 nodes. The impact of network density can be explained by the augmentation of simultaneous co-located contacts. In the same neighborhood, nodes can get pieces from more neighbors when the network in denser. In particular, two co-located nodes can get the same pieces at the same time but from different neighbors. As a consequence, a future contact between these two nodes becomes useless. We observe, however, that PACS limits neighborhood



(a) Random trip model. (b) Community-based model.

Figure 10: Nodes infection evolution. 250 nodes. Dissemination of a 48MB data divided on 128 pieces of 384kB. Area $1,000m \times 1,000m$.



(a) Random trip model. (b) Community-based model.

Figure 11: Neighborhood redundancy. Dissemination of a 48MB data divided on 128 pieces of 384kB. Area $1,000m \times 1,000m$.

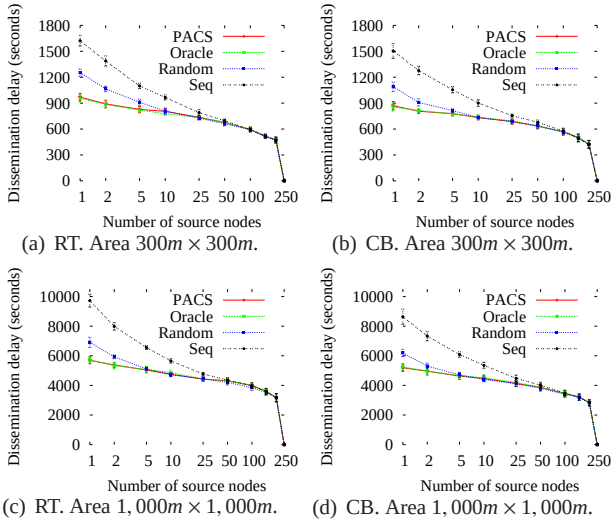
redundancy as compared to sequential and random strategies. For example, with 500 nodes, the number of useless contacts with PACS is divided by two comparing to the random strategy. This highlights the fact that co-located nodes get more heterogeneous pieces with PACS.

8.4. Impact of the number of data sources

We study the impact of the number of data sources on the dissemination delay of the different strategies (Fig. 12). Recall that a data source is a node that initially has all the pieces. As expected, for all the strategies, the dissemination delay decreases with the increase of the number of data sources in the network. Nevertheless, the input of additional sources does not give the same proportion of improvement to the overall dissemination delay. For example, when we double the number of data sources from 50 to 100, the delay decreases by less than 15% for all the strategies. Hence, the benefits of introducing new data sources decreases as the number of data sources increases. In addition, we observe that the improvement is less important for PACS and Oracle. Indeed, the delay only decreases by 20% when changing from 1 to 25 sources, while it reaches more than 40% for the random and sequential strategies. It is important to note that the results of the random and the sequential strategies become very close to the one obtained by PACS and Oracle when the number of data sources increases.

8.5. Impact of fragment dispersion

We want to figure out if the piece dispersion can impact the obtained results. Fig 13 shows the dissemination delay when the number of piece sources varies from 1 to 32. When the number of piece sources is equal to 1, all the pieces are initially available at one node. This configuration is equivalent to the single data source scenario. In contrast, when the number of



(a) RT. Area $300m \times 300m$. (b) CB. Area $300m \times 300m$.

(c) RT. Area $1,000m \times 1,000m$. (d) CB. Area $1,000m \times 1,000m$.

Figure 12: Dissemination delay according to the number of data sources. Dissemination of a 12MB data divided on 32 pieces of 384kB. Random trip (RT) versus community-based model (CB).

piece sources is equal to 32, each piece is initially available at a different node in the network. In this study, we do not consider the sequential strategy since the strategy principle does not hold anymore if more than one piece source exists. Indeed, the initial state represents nodes that did not get the pieces in an increasing order of identifiers. We only evaluate the dissemination delay for the random strategy, PACS, and Oracle. Even if the dissemination delay decreases with the increase of piece source for all strategies, the improvement achieved is more significant with the random strategy. Indeed, the delay is improved by more than 30% in this case, while it only reaches 12% with PACS and Oracle. This result can be explained by the fact that distributing different pieces to different nodes enables an initial piece shuffling. In addition to node mobility, this can be enough to get a good heterogeneity of pieces in the network even with a random piece selection strategy. However, such an initial dispersion of piece is not obvious in real content sharing scenario.

9. Real-world trace evaluation

In this section, we evaluate the performance of the spreading strategies using the real-world mobility traces of RollerNet. We vary the scenario by setting each node in the network as data source. Plots represent average results. Section 7.3 summarizes the experimentation details.

9.1. Impact of the piece size

When the piece header is set to 0, the dissemination delay increases with the augmentation of the piece size and those regardless the strategy (Fig. 14). One reason is that the larger the piece size, the less the number of contact opportunities able to transmit the piece. Moreover, when the piece is too voluminous, the dissemination fails in many cases. This is what happens when trying to send pieces larger than 1.5MB (resp.

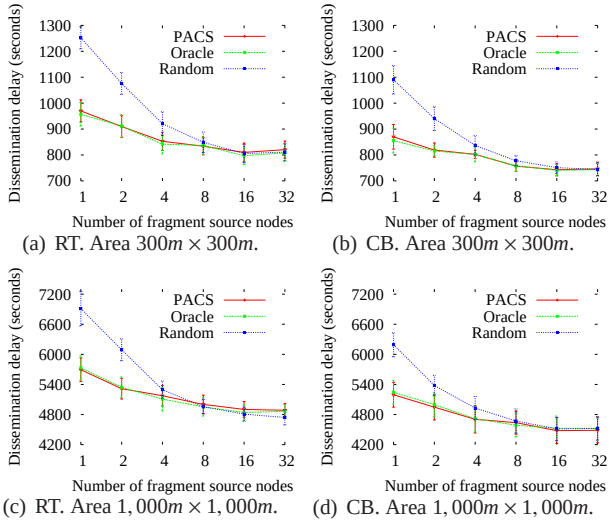


Figure 13: Dissemination delay according to the number of piece sources. Dissemination of a 12MB data divided on 32 pieces of 384kB. Random trip (RT) versus community-based model (CB).

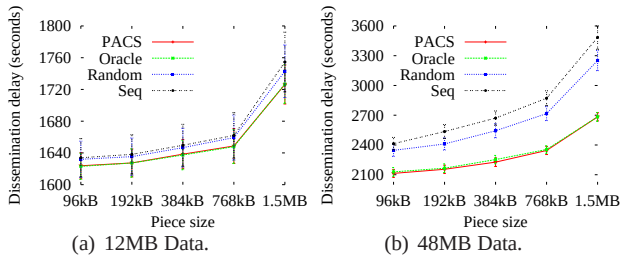


Figure 14: Dissemination delay according to the piece size. Header size set to 0. Nodes move based on RollerNet trace. (Please note that the two graphs do not use the same scale, for the sake of visualization).

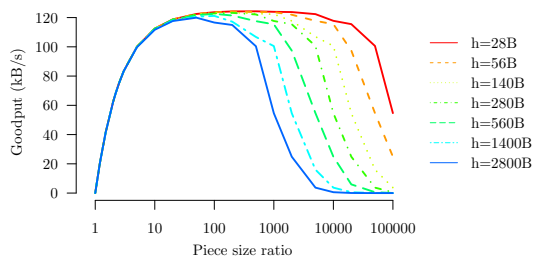


Figure 15: Global goodput according to the piece size ratio. Header size varies from 28B to 2800B. Piece size varies as a multiple of the header size. RollerNet trace.

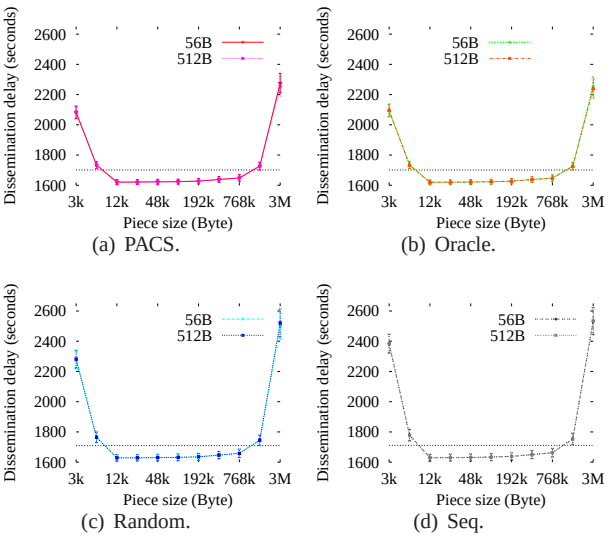


Figure 16: Dissemination delay according to the piece size when overhead is considered. Dissemination of a 12MB data. Nodes move based on RollerNet trace.

3MB) for 48MB data (resp. 12MB data). Nevertheless, comparing the different strategies, the increase of the dissemination delay is less significant with PACS than with the sequential and random strategies. This difference is more noticeable when disseminating larger data (Fig. 14(b)). Indeed, when the number of contact opportunities able to transmit the piece is smaller, the impact of the strategy grows.

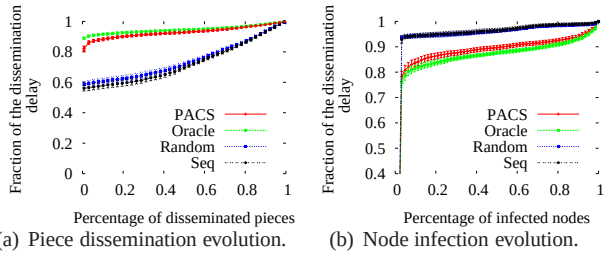
9.2. *Impact of the overhead*

We now study the impact of the overhead introduced by the piece header on the previous results. Fig. 15 shows the global goodput obtained from the RollerNet trace according to the piece size. We consider a large scale of header size from reasonable values (28B, 56B) to some very large values (1,400B, 2,800B). Although a header of 2,800B is impractical in a real deployment, we intentionally consider such large header sizes to understand the tendency of the resulting goodput. We vary the piece size according to the header size. When piece size ratio equals 1, the piece contains only the header. In this case, the goodput is equal to 0 since no useful data is sent. On the other hand, when the piece size ratio is very large, the header represents a small fraction of the piece size. However, here again, the goodput tends to 0 since the number of usable contact decreases as the piece becomes too large. Between these extreme values, the shape of the goodput curve depends on the value of the header size. The larger the header size, the sharper the goodput curve. This means that the selection of the piece size is crucial when the header is significant. Indeed, when the header is large, the goodput considerably changes depending on the piece size. In all cases, the maximum global goodput is obtained with a piece size ratio around 90. What is interesting to observe here is that the smaller the header, the larger the plateau of piece size ratios that optimize the goodput.

Plots in Fig. 16 concur with the results obtained with the theoretical computation of the global goodput. Fig. 16 shows the dissemination delay according to the piece size when the header equals 56B and 512B. For the four strategies, when the piece is in range [6kB, 768kB], we obtain similar dissemination delays. This confirms that when the header remains reasonable, we can select the piece size among a large range of values without significant consequences on the results.

9.3. *Impact of the piece selection strategy*

This section investigates the importance of the piece selection strategy in a real environment. We analyze the impact of



(a) Piece dissemination evolution. (b) Node infection evolution.

Figure 17: Dissemination evolution. Dissemination of a 48MB data divided on 128 pieces of 384kB. Nodes move based on RollerNet trace.

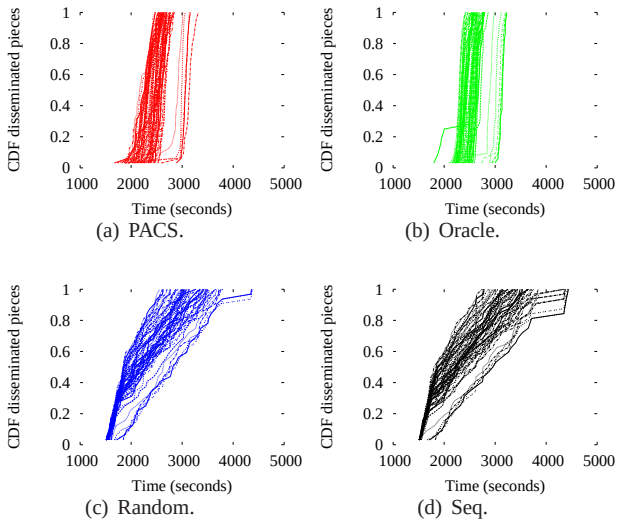
the strategy in the evolution of both piece dissemination and node infection. Figs. 17(a) and 17(b) confirm the observations made with the mobility models. Indeed, compared to the sequential and random strategies, PACS achieves slower piece dissemination and a faster node infection. Clearly, the percentage of nodes having all pieces and playing the role of a source node increases faster with PACS. This observation reflects a higher heterogeneity of the disseminated pieces with PACS that explains the better dissemination delay.

To see in detail how the dissemination evolves in time, we estimate the piece dissemination delay (Fig. 18). We define the piece dissemination delay as the time required for a particular piece to be fully disseminated. We consider the dissemination of a 48MB data divided into 32 pieces of 1.5MB. Each plot in the figures represents a different data source. We clearly distinguish two different behaviors. On the one hand, the random and the sequential strategies (Fig. 18(c), Fig. 18(d)) achieve the dissemination of the first pieces very quickly. Nevertheless, they spend much more time to disseminate the last pieces. This can be explained by the lack of piece diversity in the network that causes useless contact opportunities. On the other hand, Oracle and PACS (Fig. 18(b), Fig. 18(a)) start by spreading various pieces. This explains the slowness for the first piece to be fully disseminated. But, because nodes get different pieces, the overall dissemination is faster.

9.4. Impact of the data source

Fig. 19 shows the dissemination delay according to the node that plays the role of the data source. We assume the dissemination of a 12MB data divided into 2 pieces of 6MB each. When the strategy fails to disseminate the content before the end of the trace, the dissemination delay is set to -1. The strategies dissemination success depends on the data source. Indeed, for some data sources (for example, nodes 26 and 50), the dissemination fails regardless the strategy. Moreover, we observe some data sources that achieve the dissemination for some strategies and fail for the others (for example, nodes 44 and 47). This latter observation highlights the fact that the piece selection strategy remains important even when the number of pieces is small (here, there are only 2 pieces). Furthermore, we notice that PACS has the same delivery ratio as Oracle and outperforms the random and sequential strategies by more than 13%.

We further investigate the dissemination failures. Fig. 20 shows the node infection delay according to data source's identifier. We consider three particular data sources: 26, 44, and 47.



(a) PACS. (b) Oracle.

(c) Random. (d) Seq.

Figure 18: Piece dissemination delay. Dissemination of a 48MB data divided on 32 pieces of 1.5MB. Nodes move based on RollerNet trace.

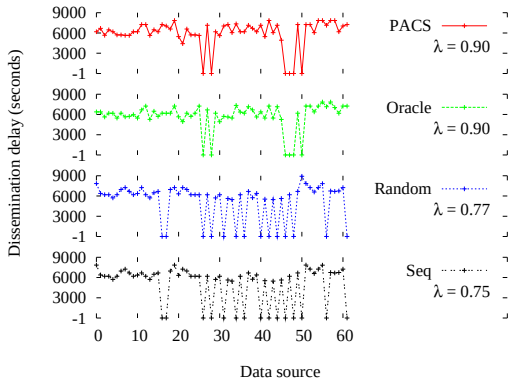
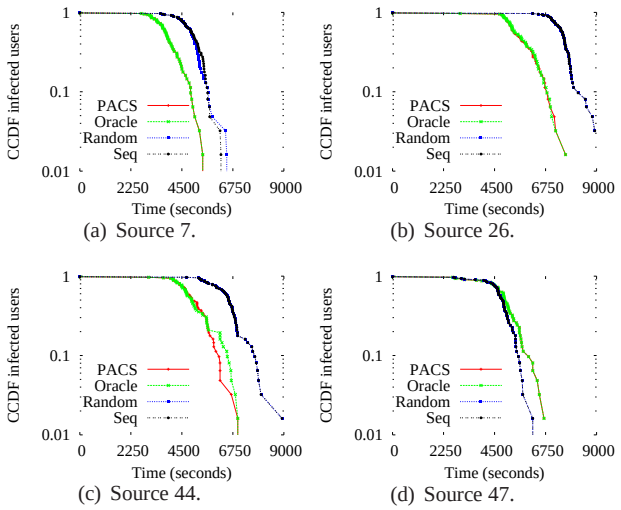


Figure 19: Dissemination delay according to the data source ID. Dissemination of a 12MB data divided on 2 pieces of 6MB. A dissemination delay equals to -1 means that the strategy fails to disseminate the content. λ is the complete delivery rate. Nodes move based on RollerNet trace.

When node 26 is the data source, no strategy completes the dissemination (this represents 8% of the points in Fig 19). In this case, the infection of the first node in the network comes very late comparing to the common case represented by the source node 7 (Fig 20(a)). Nevertheless, even if the dissemination is not achieved for all strategies, the node infection delay is faster in PACS comparing to the random and sequential strategies. Indeed, with PACS, 95% of nodes are infected at the time 7,135 of the trace whereas the same rate is reached by the random and sequential strategies at time 8,810. When node 44 is the source, PACS and Oracle complete the dissemination while the random and sequential strategies fail (represents 14.5% of the points in Fig 19). Here, the random and sequential strategies infect only 82% nodes when PACS achieves full dissemination. Finally, when node 47 is the source, random and sequential strategies achieve the dissemination while Oracle and PACS fail (represents 1.6% of the points in Fig. 19). In this case, PACS infects 98% of the nodes at time 6,681 and fails to infect the last node



(a) Source 7. (b) Source 26. (c) Source 44. (d) Source 47.

Figure 20: Node infection delay according to data source ID. Dissemination of a 12MB data divided on 2 pieces of 6MB. Nodes move based on RollerNet trace.

even if it still remains 30% of the total time. We find that the last non-infected node becomes isolated at this moment. This is due to the random selection of the neighbor with whom pieces are exchanged.

10. Conclusion and open issues

In this paper, we investigate challenges of large content dissemination in opportunistic networks. First, we observe that fragmentation reduces the dissemination delay and even enables the dissemination of contents that could not be disseminated otherwise. Second, we address the piece size selection problem and show that, for reasonable header size, the piece size can in general be selected from a large range of values without significant impact on the results. Finally, we proposed, designed, and evaluated PACS, an efficient strategy to disseminate large contents in opportunistic networks. PACS selects pieces to disseminate based on a local view of their dissemination progress in the network. We evaluate PACS using both mobility models and real-world trace simulations. Thanks to higher heterogeneity when distributing pieces, PACS achieves better dissemination delays and faster node infection than the sequential and random strategies.

Future directions include a number of open issues. A first question is the impact of the selected neighbor, i.e., how to better select the relaying node when having several simultaneous contact opportunities. Results from Section 9 indicate that the random selection of the relaying nodes can lead to situations where the last non-infected node becomes isolated. A more sophisticated relay selection strategy could avoid such a scenario by infecting nodes likely to be isolated first. Second, we could extend the population of users to the case of multiple groups of various interests. In this case, extending the algorithm with a caching policy could be a good solution [29, 30]. Finally, we also intend to extend PACS with networking coding capabilities.

Acknowledgment

This work is partially supported by the ANR Crowd project under contract ANR-08-VERS-006.

References

[1] J. LeBrun, C. N. Chuah, Bluetooth content distribution stations on public transit, in: International workshop on Decentralized resource sharing in mobile computing and networking, Los Angeles, USA.

[2] V. Lenders, G. Karlsson, M. May, Wireless Ad Hoc Podcasting, in: IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks.

[3] L. McNamara, C. Mascolo, L. Capra, Media sharing based on colocation prediction in urban transport, in: ACM Mobicom, San Francisco, USA.

[4] P. Gill, M. Arlitt, Z. Li, A. Mahanti, YouTube traffic characterization: A view from the edge, in: ACM IMC, San Diego, USA.

[5] P. U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. D. de Amorim, J. Whitbeck, The accordion phenomenon: Analysis, characterization, and impact on DTN routing, in: IEEE Infocom, Rio de Janeiro, pp. 1116–1124.

[6] S. Gaito, E. Pagani, G. P. Rossi, Opportunistic forwarding in workplaces, in: ACM SIGCOMM workshop on Online Social Networks, Barcelona, Spain.

[7] A. Barrat, C. Cattuto, V. Colizza, J.-F. Pinton, W. V. den Broeck, A. Vespignani, High resolution dynamical mapping of social interactions with active RFID, 2008. ArXiv:0811.4170.

[8] A. Chaintreau, P. Hui, C. Diot, R. Gass, J. Scott, Impact of human mobility on opportunistic forwarding algorithms, IEEE Transactions on Mobile Computing 6 (2007) 606–620. Fellow-Jon Crowcroft.

[9] N. Eagle, A. S. Pentland, Reality mining: Sensing complex social systems, Personal Ubiquitous Computing 10 (2006) 255–268.

[10] N. Belblidia, M. Dias de Amorim, L. H. M. K. Costa, J. Leguay, V. Conan, PACS: Chopping and shuffling large contents for faster opportunistic dissemination, in: International Conference on Wireless On-Demand Network Systems and Services, Bardonecchia, Italy.

[11] S. PalChaudhuri, J. Y. Le Boudec, M. Vojnovic, Perfect simulations for random trip mobility models, in: IEEE Annual Simulation Symposium, San Diego, USA.

[12] M. Musolesi, C. Mascolo, A community based mobility model for Ad Hoc network research, in: ACM Realman, Florence, Italy.

[13] B. Williams, T. Camp, Comparison of broadcasting techniques for mobile ad hoc networks, in: ACM Mobihoc, Lausanne, Switzerland, pp. 194–205.

[14] C. Fragouli, J. Widmer, J.-Y. L. Boudec, Efficient broadcasting using network coding, IEEE/ACM Transactions on Networking 16 (2008) 450–463.

[15] A. Asterjadhi, E. Fasolo, M. Rossi, J. Widmer, M. Zorzi, Toward network coding-based protocols for data broadcasting in wireless Ad Hoc networks, IEEE Transactions on Wireless Communications 9 (2010).

[16] M. Pitkänen, A. Keränen, J. Ott, Message fragmentation in opportunistic dtns, in: IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, Newport Beach, CA, USA.

[17] S. Rajagpalan, C. C. Shen, A cross-layer decentralized BitTorrent for mobile Ad Hoc networks, in: International Conference on Mobile and Ubiquitous Systems: Networks and Services, San Jose, USA.

[18] N. Gaddam, A. Potluri, Study of BitTorrent for file sharing in Ad Hoc networks, in: IEEE Wireless Communications and Networking Conference, Allahabad, India.

[19] M. Sbai, E. Salhi, C. Barakat, P2P content sharing in spontaneous multihop wireless networks, in: International conference on Communication systems and Networks, Bangalore, India.

[20] A. Nandan, S. Das, G. Pau, M. Gerla, M. Sanadidi, Co-operative downloading in vehicular Ad Hoc wireless networks, in: Annual Conference on Wireless On-demand Network Systems and Services.

[21] S. Goel, M. Singh, D. Xu, B. Li, Efficient peer-to-peer data dissemination in mobile Ad Hoc networks, in: International Conference on Parallel Processing Workshops.

[22] U. Lee, S. Jung, D.-K. Cho, A. Chang, J. Choi, M. Gerla, P2P content distribution to mobile bluetooth users, IEEE Transactions on Vehicular Technology 59 (2010) 356 –367.

[23] S. Jung, U. Lee, A. Chang, D.-K. Cho, M. Gerla, Bluetorrent: Cooperative content sharing for bluetooth users, in: IEEE International Conference on Pervasive Computing and Communications, White Plains, New York, USA.
[24] O. R. Helgason, E. A. Yavuz, S. T. Kouyoumdjieva, L. Pajevic, G. Karlsson, A mobile peer-to-peer system for opportunistic content-centric networking, in: ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds, New Delhi, India, pp. 21–26.
[25] B. Cohen, Incentives build robustness in BitTorrent, in: P2PEcon, Berkeley, USA.
[26] BitTorrent protocol specification v1.0, ????
[27] R. Xia, J. Muppala, A survey of bittorrent performance, IEEE Communications Surveys and Tutorials 12 (2010) 140–158.
[28] A. Keränen, J. Ott, T. Kärkkäinen, The ONE Simulator for DTN Protocol Evaluation, in: International Conference on Simulation Tools and Techniques, Rome, Italy.
[29] L. Yin, G. Cao, Supporting cooperative caching in Ad Hoc networks, IEEE Transactions on Mobile Computing 5 (2006) 77–89.
[30] M. Fiore, F. Mininni, C. Casetti, C. Chiasserini, To Cache or Not To Cache?, in: IEEE Infocom, Rio de Janeiro, Brazil.