PACS: Chopping and shuffling large contents for faster opportunistic dissemination

Nadjet Belblidia, Marcelo Dias de Amorim, Luís Henrique Maciel Kosmalski Costa, Jeremie Leguay, Vania Conan

PACS: Chopping and shuffling large contents for faster opportunistic dissemination

Nadjet Belblidia*, Marcelo Dias de Amorim*, Luís Henrique M. K. Costa†, Jérémie Leguay‡, and Vania Conan‡

* UPMC Sorbonne Universités, Paris, France
† Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil
‡ Thales Communications, Colombes, France

***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 nodes produce contents that are larger than the capacity of a contact. In such a case, nodes must slice data and send fragments separately, which allows better use of short contacts and progressive dissemination of large contents data pieces. The question here is to design the best strategy for deciding which piece(s) to transmit whenever nodes meet. In this paper, 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 version based on an oracle.**

I. 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 the median contact time is 48 seconds, i.e., more than 50% of the contacts last for less than 1 minute. 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 size of messages to the contact capacity by fragmenting the transmitted data. In this paper, we suppose that contents are divided into pieces of standard size. The main challenge when disseminating fragmented data is to decide which piece(s) should be sent when two nodes meet (see Section III). 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. 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.

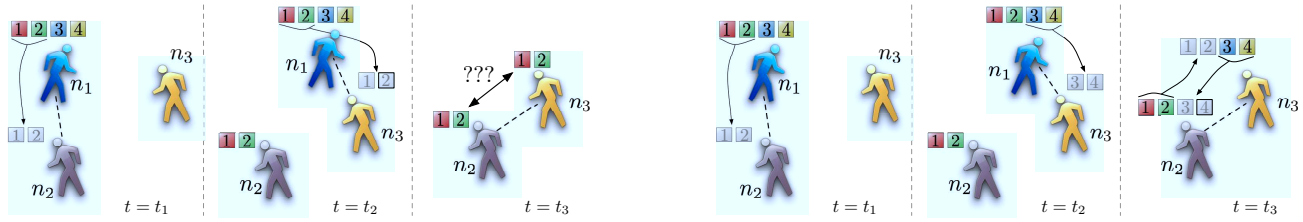
To counterpart the abovementioned issues, we propose PACS (Prevalence-Aware Content Spreading), a popularity-based strategy to select pieces to be exchanged between neighbors solely based on node-local information. Through their successive contacts, nodes keep track of the dissemination level of the 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 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 [10] and the community-based mobility model proposed in [11]. Additionally, we have also used movement traces obtained from RollerNet, an intermittently-connected mobile network formed between 62 people during a rollerblading tour in the streets of Paris, which lasts for 3 hours [5].

In summary, the key contributions of PACS are:

- **Higher heterogeneity of pieces in the network.** PACS

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



(a) Sequential. (b) Random.

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

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.

II. Content Spreading in Opportunistic Networks

In this section, we provide all the necessary background before introducing the dissemination algorithms.

A. Problem Statement

In our problem, we consider a certain number of *pieces* that compose a *content*. The content must be disseminated to a population of mobile nodes that communicate opportunistically. The problem we face here is: *given the pieces and a contact opportunity, which subset of these pieces should be transferred if the contact is insufficient to transmit all of them?*

Our objective is to define an algorithm to select pieces to be transferred when two nodes get into communication range. This algorithm should generate little overhead and still lead to fast content dissemination. Note that this problem is similar to file swarm in peer-to-peer networks [12]. However, in our case infection can only happen when nodes meet physically.

B. Motivating Example

We now illustrate why the proper selection of pieces to send is important. The straightforward approach for a node to disseminate a 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 illustrated 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. We propose a generalization to the solution shown in Fig. 1(b).

TABLE I
Summary of the variables.

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

C. 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 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. 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 I.

III. Basic content dissemination strategies

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

A. 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 node n_i has, i.e., $\hat{c}_{n_i} = j$ if $a_j = 1$ and $a_{j+1} = 0$. 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 $\hat{c}$.

Note that if $\hat{c}_{n_i} = \hat{c}_{n_j}$, the contact will be useless as the nodes have exactly the same contents. For a contact of duration t, the maximum number of pieces transferred is $\min\{|\hat{c}_i - \hat{c}_j|; \lfloor t/\tau \rfloor\}$.

B. 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 II-C). 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 time, 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:

$$\begin{aligned} \mathbf{a}_{n_i} &= \mathbf{a}_{n_i} \wedge \mathbf{i}_{c_{j\to i}}, \\ \mathbf{a}_{n_j} &= \mathbf{a}_{n_j} \wedge \mathbf{i}_{c_{i\to j}}. \end{aligned} \tag{1}$$

where $\mathbf{i}_{c_{i\to j}}$ and $\mathbf{i}_{c_{j\to i}}$ 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.

IV. 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 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:

$$\begin{aligned} \mathbf{p}_{n_i} &= \mathbf{p}_{n_i} + \mathbf{a}_{n_j}, \\ \mathbf{p}_{n_j} &= \mathbf{p}_{n_j} + \mathbf{a}_{n_i}. \end{aligned} \tag{2}$$

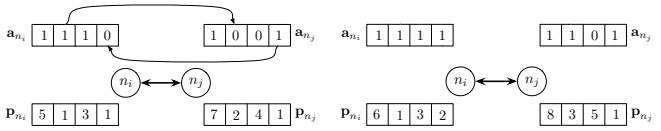
Among the candidate pieces to be transferred, nodes select the one with the lowest prevalence. In case of tie, a piece is chosen in a uniformly distributed random way. Let $c_{i\to j}$ be the piece sent by n_i to n_j and $c_{j\to i}$ be the piece sent by n_j to n_i. Once this step done, nodes update their availability vectors as indicated in Equation 1.

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. 2. After exchanging their availability vectors, nodes update their prevalence vectors as indicated in Equation 2 ($\mathbf{p}_{n_i} = \{6, 1, 3, 2\}$ and $\mathbf{p}_{n_j} = \{8, 3, 5, 1\}$). 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 1.

Note that PACS has some similarities with peer-to-peer systems, notably BitTorrent [12], [13]. 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

Algorithm 1 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 ← prevalence_selection_from((a_{n_i} ∧ (¬a_{n_j})), p_{n_i});
6:       send_to(n_j, c_s);
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);
10:      i_{c_{j→i}} = {i_0, ..., i_{K-1}}; i_k = 0, ∀k < K (k ≠ s) and i_s = 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.

Fig. 2. Piece selection using PACS. Initially, n_i has pieces $\{c_0, c_1, c_2\}$ and n_j has pieces $\{c_0, c3\}$.

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.

V. Evaluation framework

In this section, we summarize the simulation and model parameters. We use the ONE [14] simulator with both mobility models and real movement trace based simulations.

A. Simulation parameters

We study the impact of the following main parameters:
Area size. We select two areas: 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 city neighborhood.
Number of nodes. The number of nodes varies between 100 and 2,500 nodes. 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.
Data size. We consider a unique content originally available at a single data source. The content size is set to either 12MB or 48MB. We select 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 96kB to 12MB. By default, the piece size is set to 384kB. The piece size together with the content size determines the number of pieces.

These parameters are summarized in Table II. Bold values stand for the defaults.

B. 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 [10]. Second, nodes move according to the community-based model formulated by Musolesi et al. [11].

For both models, nodes move at walking speed (between 0.5m/s and 1.5m/s). Two nodes are able to communicate when in communication range of 10m. Data is transferred at a throughput of 125 kBps. In addition, each model has specific parameters. For random trip, nodes may pause between two trips. Node pause time is uniformly selected 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 II summarizes the parameters of the models.

TABLE II
Simulation parameters.

Factors	Area size	300m×300m, 1,000m×1,000m
	Number of nodes	100, **250**, 500, 1,000, 2,500
	Data size	12MB, 48MB
	Piece size	96kB, 192kB, **384kB**, 768kB, 1.5MB, 3MB, 6MB, 12MB
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

C. 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.[2]

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 125 kBps that correspond to an average Bluetooth throughput. At each simulation run, the data source is changed. The trace configuration is summarized in Table II.

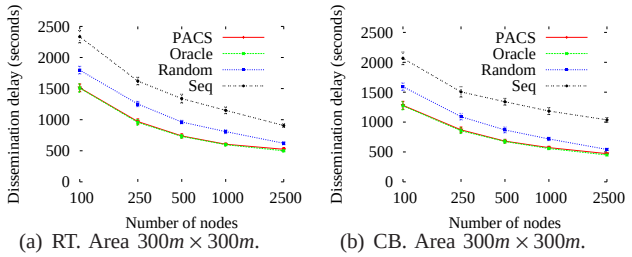
D. Benchmarking

We compare PACS with the two basic strategies described in Section III: the sequential strategy and the uniform random strategy. Besides these strategies, we 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 dissemination *Oracle*. Obviously, deploying such a centralized strategy is impracticable in a real opportunistic network. We only use it for comparison purposes.

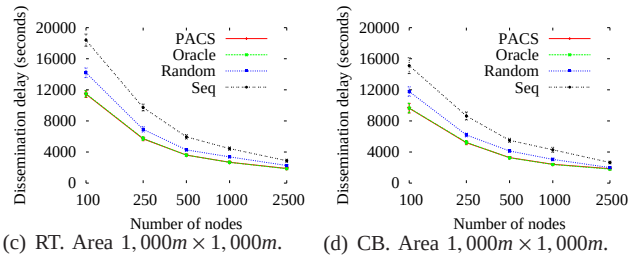
VI. Synthetic mobility evaluation

We use two mobility models to generate synthetic traces. First, we study the simple case of mobility induced by the

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

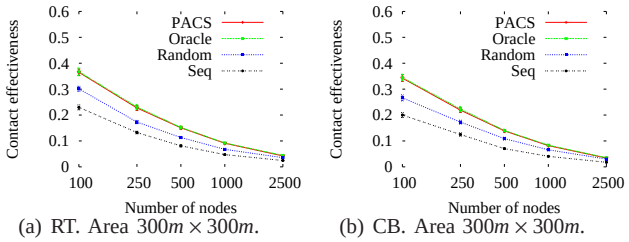


(a) RT. Area $300m \times 300m$. (b) CB. Area $300m \times 300m$.

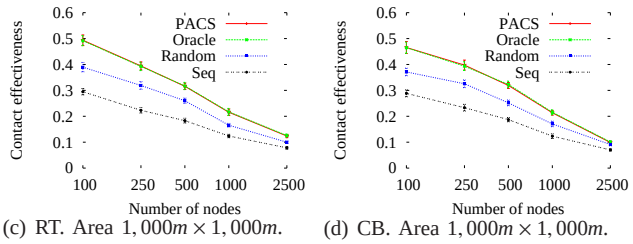


(c) RT. Area $1,000m \times 1,000m$. (d) CB. Area $1,000m \times 1,000m$.

Fig. 3. 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 $1,000m \times 1,000m$.

Fig. 4. 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).

random trip model. Second, we consider the community-based mobility model, a more elaborated model founded on social network theory. Parameter details are in Section V-B.

A. Impact of the network density and the network diameter

We vary both the area size and the number of nodes to study the dissemination delay of a 12MB-file (Fig. 3). 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. 4). The contact effectiveness is the ration between 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 gives the worse performances. Such a tendency is asserted in dense networks (Fig. 3(a) and Fig. 3(b)). Regardless of the number of nodes and the area size, PACS performs better than the sequential and the random strategies, achieving more than 2× faster dissemination delays. Furthermore, the results of PACS tend to the ones obtained using the oracle strategy.

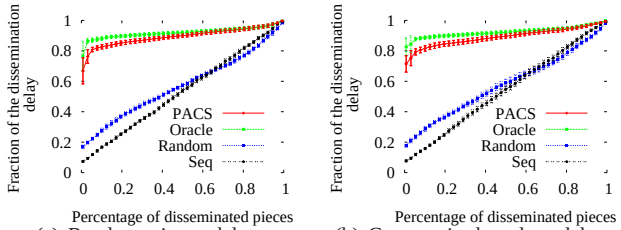
B. Impact of the strategy on the piece dissemination evolution

We want to understand the reason of such a difference in the dissemination delay between the strategies. First, we compare the strategies regarding the piece dissemination evolution. Fig. 5 shows the proportion of time required, among the total time, to fully disseminate a specific 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. In opposite, with PACS and oracle, nodes start by getting different pieces and no piece is fully disseminated before 82% of the total time for Oracle and 71% of the total time for PACS.

Then, we analyze the impact of the strategy on the node infection evolution. Fig. 6 shows the proportion of time required, among the total time, to infect a definite percentage of nodes. A node is infected when getting 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 at the middle 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. Indeed, in a real scenario, we get more satisfied nodes with PACS since the node infection is faster.

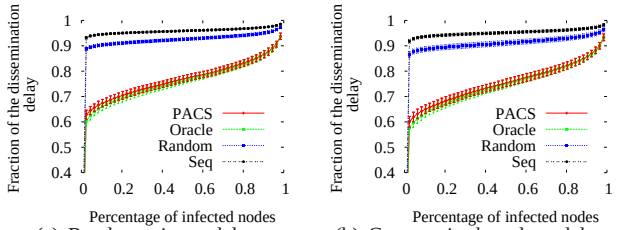
C. Impact of the strategy on the neighborhood redundancy

We define the neighbor redundancy as the average fraction of useless connections selected by each node at each slot. A connection is considered useless if the two nodes involved in it have no pieces to exchange. We consider the dissemination of a 48MB file divided on 128 pieces. Fig 7 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



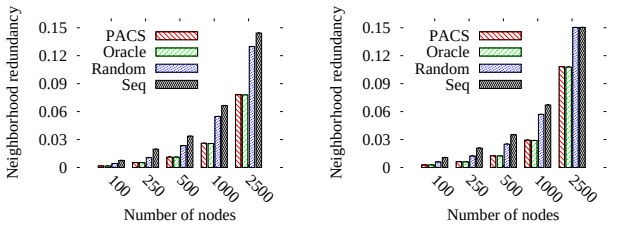
(a) Random trip model. (b) Community-based model.

Fig. 5. Piece dissemination evolution. 250 nodes. Dissemination of a 48MB data divided on 128 pieces of 384kB. Area $1,000m \times 1,000m$.



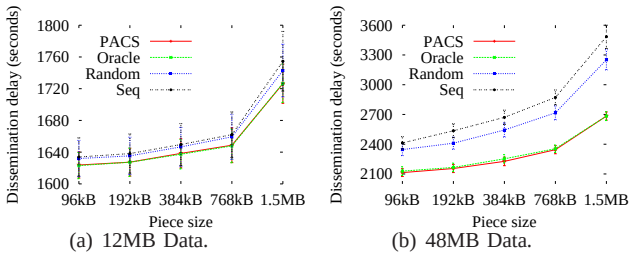
(a) Random trip model. (b) Community-based model.

Fig. 6. 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.

Fig. 7. Neighborhood redundancy. Dissemination of a 48MB data divided on 128 pieces of 384kB. Area $1,000m \times 1,000m$.



(a) 12MB Data. (b) 48MB Data.

Fig. 8. Dissemination delay according to the piece size. Nodes move based on RollerNet trace. (Please note that the two graphs do not use the same scale, for the sake of visualization.)

random trip model for example (Fig. 7(a)), only 1% or less of the selected connection is useless with 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 redundancy as compared to sequential and random strategies. For example, with 500 nodes, the number of useless connection 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.

VII. Real-world trace evaluation

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

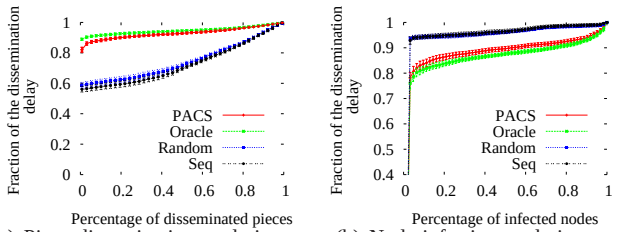
A. Impact of the piece size

Regardless the strategy, the dissemination delay increases with the augmentation of the piece size (Fig. 8). 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. 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. 8(b)). Indeed, when the number of contact opportunities able to transmit the piece is smaller, the impact the strategy grows.

B. Impact of the piece selection strategy

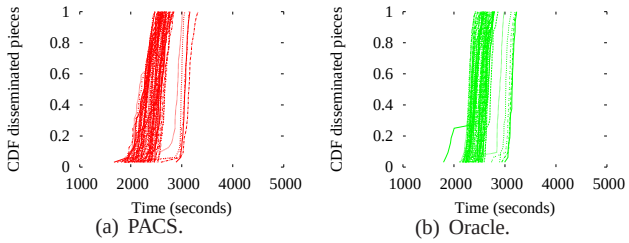
This section investigates the importance of the piece selection strategy in a real environment. We analyze the impact of the strategy in the evolution of both piece dissemination and node infection. Figs. 9(a) and 9(b) confirm the observations made with the mobility models. Indeed, compared to the sequential 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. 10). 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. 10(c), Fig. 10(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. 10(b), Fig. 10(a)) start by spreading various pieces. This explains the slowness of the first piece dissemination. But, because nodes get different pieces, the overall dissemination is faster.

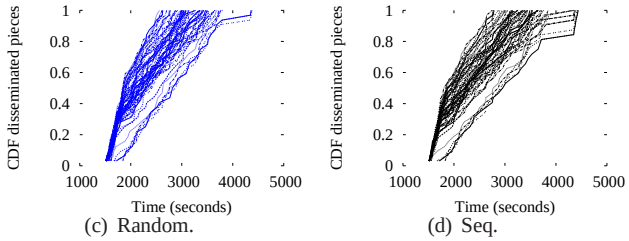


(a) Piece dissemination evolution. (b) Node infection evolution.

Fig. 9. Dissemination evolution. Dissemination of a 48MB data divided on 128 pieces of 384kB. Nodes move based on RollerNet trace.



(a) PACS. (b) Oracle.



(c) Random. (d) Seq.

Fig. 10. Piece dissemination delay. Dissemination of a 48MB data divided on 32 pieces of 1.5MB. Nodes move based on RollerNet trace.

C. Impact of the data source

The data source may have an impact on the dissemination success. Fig. 11 shows the dissemination delay according to the data source ID. We assume the dissemination of a 12MB data divided on 2 pieces of 6MB. 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 last observation highlights the fact that the piece selection strategy remains important even when the number of pieces is small (here, there is only 2 pieces). Furthermore, we notice that PACS has the same delivery rate as oracle and outperforms the random and sequential strategies by more than 13%.

We further investigate the dissemination failures. Fig. 12 shows the node infection delay according to data source ID. We denote the node infection delay as the elapsed time before a particular node receives the full content. We consider three particular data sources: 26, 44, and 47. When node 26 is the data source, no strategy completes the dissemination (this represents 8% of the points in Fig 11). In this case, the

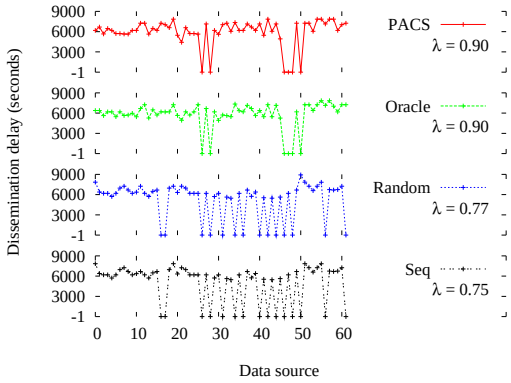


Fig. 11. Dissemination delay according to the data source ID. Dissemination of a 12MB data divided on 2 pieces of 6MB. A dissemination delay equals to -1 means that the strategy fails to disseminate the content. λ is the complete delivery rate. Nodes move based on RollerNet trace.

infection of the first node in the network comes very late comparing to the common case represented by the source node 7 (Fig 12(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 11). Here, the random and sequential strategies infect only 82% nodes when PACS achieves the 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. 11). In this case, PACS infects 98% of the nodes at time 6,681 and fails to infect the last node even if it 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.

VIII. Related work

As discussed in Section IV, our solution is inspired by BitTorrent. Several solutions have been proposed to adapt BitTorrent to opportunistic and Ad Hoc networks [15], [16], [17]. 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 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 [18]. 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 prioritize rarer pieces and consider peer location. SPAWN

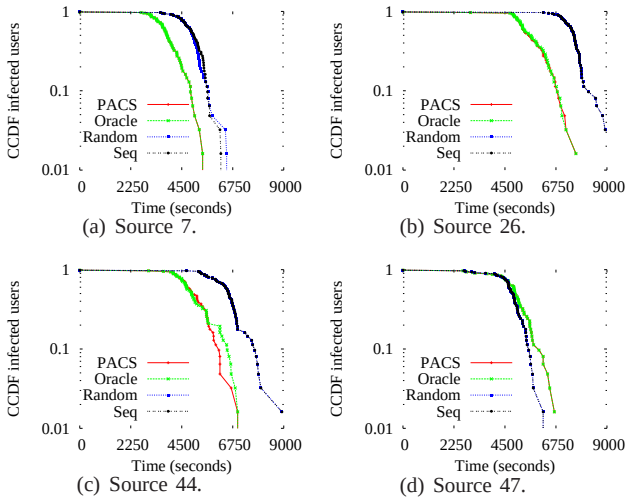


Fig. 12. Node infection delay according to data source ID. Dissemination of a 12MB data divided on 2 pieces of 6MB. Nodes move based on RollerNet trace.

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 a routing protocol that maintains multi-hop routes between peers.

Some other solutions implemented file swarming by only considering one-hop communications [19], [20]. 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.

IX. Conclusion and open issues

In this paper, we proposed, designed, and evaluated PACS, an efficient strategy to disseminate large contents in opportunistic networks. PACS selects pieces to disseminate based on their prevalence in the network. We evaluate PACS using both mobility based and real-world trace simulations. Thanks to a more heterogeneous piece distribution, PACS achieves better dissemination delays and faster node infection than the sequential and random strategies.

Future work includes several interesting 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. Second, in this paper, we considered a unique content dissemination. An interesting future direction is to consider the case of multiple contents to disseminate to multiple user groups. In this case, extending the algorithm with a caching policy could be a good solution [21], [22]. Another important concern is the overhead induced by content fragmentation in this scenario (which includes extra headers for each piece). Indeed, our results showed that faster dissemination is obtained with smaller pieces. Nevertheless, to implement such a solution in reality, we must account for the tradeoff between overhead and dissemination delay. 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, the European Commission in the framework of the FP7 Network of Excellence NEW-COM++ under contract 216715, and the following Brazilian Agencies: Capes, CNPq, Faperj, and Finep/Funttel.

References

[1] J. LeBrun and 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, 2006.
[2] V. Lenders, G. Karlsson, and M. May, “Wireless Ad Hoc Podcasting,” in *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2007.
[3] L. McNamara, C. Mascolo, and L. Capra, “Media sharing based on colocation prediction in urban transport,” in *ACM Mobicom*, San Francisco, USA, 2008.
[4] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, “YouTube traffic characterization: A view from the edge,” in *ACM IMC*, San Diego, USA, 2007.
[5] P. U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. D. de Amorim, and J. Whitbeck, “The accordion phenomenon: Analysis, characterization, and impact on DTN routing,” in *IEEE Infocom*, Rio de Janeiro, 2009.
[6] S. Gaito, E. Pagani, and G. P. Rossi, “Opportunistic forwarding in workplaces,” in *ACM SIGCOMM workshop on Online Social Networks*, Barcelona, Spain, 2009.
[7] A. Barrat, C. Cattuto, V. Colizza, J.-F. Pinton, W. V. den Broeck, and 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, and J. Scott, “Impact of human mobility on opportunistic forwarding algorithms,” *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 606–620, 2007, fellow-Jon Crowcroft.
[9] N. Eagle and A. S. Pentland, “Reality mining: Sensing complex social systems,” *Personal Ubiquitous Computing*, vol. 10, no. 4, pp. 255–268, 2006.
[10] S. PalChaudhuri, J. Y. Le Boudec, and M. Vojnovic, “Perfect simulations for random trip mobility models,” in *IEEE Annual Simulation Symposium*, San Diego, USA, 2005.
[11] M. Musolesi and C. Mascolo, “A community based mobility model for Ad Hoc network research,” in *ACM Realman*, Florence, Italy, 2006.
[12] B. Cohen, “Incentives build robustness in BitTorrent,” in *P2PEcon*, Berkeley, USA, 2003.
[13] BitTorrent protocol specification v1.0. [Online]. Available: http://wiki.theory.org/BitTorrentSpecification
[14] A. Keränen, J. Ott, and T. Kärkkäinen, “The ONE Simulator for DTN Protocol Evaluation,” in *International Conference on Simulation Tools and Techniques*, Rome, Italy, 2009.
[15] S. Rajagpalan and 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, 2006.
[16] N. Gaddam and A. Potluri, “Study of BitTorrent for file sharing in Ad Hoc networks,” in *IEEE Wireless Communications and Networking Conference*, Allahabad, India, 2009.
[17] M. Sbai, E. Salhi, and C. Barakat, “P2P content sharing in spontaneous multi-hop wireless networks,” in *International conference on Communication systems and Networks*, Bangalore, India, 2010.
[18] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Sanadidi, “Co-operative downloading in vehicular Ad Hoc wireless networks,” in *Annual Conference on Wireless On-demand Network Systems and Services*, 2005.
[19] S. Goel, M. Singh, D. Xu, and B. Li, “Efficient peer-to-peer data dissemination in mobile Ad Hoc networks,” in *International Conference on Parallel Processing Workshops*, 2002.
[20] U. Lee, S. Jung, D.-K. Cho, A. Chang, J. Choi, and M. Gerla, “P2P content distribution to mobile bluetooth users,” *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 356 –367, 2010.
[21] L. Yin and G. Cao, “Supporting cooperative caching in Ad Hoc networks,” *IEEE Transactions on Mobile Computing*, vol. 5, no. 1, pp. 77–89, 2006.
[22] M. Fiore, F. Mininni, C. Casetti, and C. Chiasserini, “To Cache or Not To Cache?” in *IEEE Infocom*, Rio de Janeiro, Brazil, 2009.