Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 3247903, 12 pages
http://dx.doi.org/10.1155/2016/3247903

Research Article

AirCache: A Crowd-Based Solution for Geoanchored Floating Data

Armir Bujari and Claudio Enrico Palazzi

Department of Mathematics, University of Padua, Padova, Italy

Correspondence should be addressed to Claudio Enrico Palazzi; cpalazzi@math.unipd.it

Received 10 June 2016; Accepted 1 August 2016

Academic Editor: Carlos T. Calafate

The Internet edge has evolved from a simple consumer of information and data to eager producer feeding sensed data at a societal scale. The crowdsensing paradigm is a representative example which has the potential to revolutionize the way we acquire and consume data. Indeed, especially in the era of smartphones, the geographical and temporal *scopus* of data is often local. For instance, users' queries are more and more frequently about a nearby object, event, person, location, and so forth. These queries could certainly be processed and answered locally, without the need for contacting a remote server through the Internet. In this scenario, the data is alimented (sensed) by the users and, as a consequence, data lifetime is limited by human organizational factors (e.g., mobility). From this basis, data survivability in the Area of Interest (AoI) is crucial and, if not guaranteed, could undermine system deployment. Addressing this scenario, we discuss and contribute with a novel protocol named AirCache, whose aim is to guarantee data availability in the AoI while at the same time reducing the data access costs at the network edges. We assess our proposal through a simulation analysis showing that our approach effectively fulfills its design objectives.

1. Introduction

The technological revolution is palpable now more than ever as we delve into the era of the Internet of Everything (IoE). Through the IoE paradigm, every object, social network profile, person, interest, and process will be part of an interconnected scenario so as to create new intelligence and services [1]. Our lives have already been enriched by the success of smartphones which have brought into our pockets any sort of sensors. Every user is now a producer of a great quantity of data that, when combined, generate new information that can then be redistributed to others. In essence, every user becomes both a producer and a consumer of information [2]; even more, data generation and consumption are two inextricable processes as the latter also provides information about user preferences, location, interest, actions, and so forth (i.e., data generation). This people-centric sensing paradigm leads to a scenario with a diminishing presence of centralized server-based services [3]. Rather, digital services will be more and more based on crowdsensing and the ability to share the information in a distributed, decentralized way [4].

In this context, it is interesting to notice how in recent years, despite the continuous improvements of the connectivity means, users have been more and more looking for location-based services. A person connects to the Internet to find out local events and restaurants, friends in proximity, touristic information, offers, news, and much more related to the environment she/he is immersed in. It could be hence more efficient to just let these kinds of information float in a certain Area of Interest (AoI), while users generate, consume, and maintain them in the AoI. The latter, in particular, could be achieved by having the AoI-related data replicated among some of the users moving in the AoI and stationing there and removed from storage if a user walks out of the AoI. In our case study, we assume that the data is initially generated by one of the users or brought from elsewhere. Nodes in the AoI can exchange data (in an opportunistic way, when in proximity of each other) for two main reasons: (i) in case one user is interested in the available floating data and

(ii) as nodes with the floating data in their storage may exit the AoI or be simply turned off, the data need sometimes to be replicated among the nodes to ensure the survivability in the AoI [5].

This creates a sort of a layer of data which is supported by the presence and participation of users and that is strictly connected to a specific AoI, hence the name *floating data* [6]. Clearly, the resilience of this system depends on the number of users that circulate at any time in an AoI but it could be certainly effective in crowded areas of any town. Furthermore, the data could appear and vanish considering the actual interest they could raise and their temporal and geographical relevance. This is in contrast with the current data model, being available globally and endlessly regardless of their consumption profile [7].

In such a system, the main features that need to be provided are accessibility and survivability in the AoI. One naive, simple but not efficient, way to generate floating data could be that of replicating the data on all the nodes in the network through data sharing at any node encounter in the AoI. Although this strategy could increase the chances for the survivability of some data in the AoI, it is also limiting and inefficient as it will consume more storage and energy than actually needed.

Addressing the issues, we discuss AirCache (AC), a distributed protocol aimed at guaranteeing data survivability in the AoI. AirCache carefully replicates the content into the network by taking into consideration energy and storage availability of nodes. Nodes in our system periodically and autonomously send beacon messages announcing their respective capabilities and content they possess. This information is then exploited in order to enforce the replication policy required to augment data availability in the AoI. Furthermore, AC is capable of controlling the spatial distribution of the data, easing data access at the network edges. The contribution of this paper is hence twofold: (i) presenting the concept of AC, its *modus operandi,* and applicability; (ii) proposing possible algorithms and protocols that can be used to generate and maintain an AC in certain locations.

The outline of the paper is as follows: in Section 2, we provide the necessary background information needed to better comprehend the context under scrutiny. Section 3 discusses state-of-the art proposals in the context of mobile networks addressing a problem similar to ours. Next, through Section 4 we introduce our proposal, discussing its *modus operandi* and key built-in features required to satisfy our goals. Section 5 discusses the simulation environment and the evaluation strategy adopted to assess the proposal, while Section 6 discusses the evaluation outcome. Finally, Section 7 concludes the paper.

2. Background

The advances on wireless technology have created many new possibilities for communication, giving rise to new and decentralized networking paradigms such as the Mobile Ad Hoc Networks (MANETs [8]). Nodes in this context dynamically self-organize in a wireless overlay providing networking support for the upper layers. This networking paradigm is inherently decentralized and communication in these settings follows the well-known peer-to-peer (P2P) communication model rather than a client-server one. In these settings, networking functions must be designed to contemplate for disruptions due to mobility [9, 10].

In contrast with MANETs, the OPPortunistic NETworking (OppNet [11]) paradigm in addition exploits unplanned communication opportunities to move data between endpoints. In MANETs mobility is a disruptive force hindering networking performance, while OppNets leverage on mobility in order to create additional communication opportunities. To achieve this goal, networking functions need to contemplate for a high degree of heterogeneity of the entities involved in communication in terms of capabilities and mobility dynamics [12, 13].

Nevertheless, the critical mass for opportunistic communications and the ever increasing device capabilities are two important factors in favor of this networking paradigm. A representative example embracing this opportunity is the people-centric sensing (crowdsensing) concept. The pervasive coverage of mobile sensing devices producing and collecting sensory data has provided the basis for new applications and services which can benefit our community as a whole [14]. Indeed, cooperative environmental monitoring or public sensing has created a cost-effective way of collecting a vast variety of data which could be exploited in several ways.

In this spirit, Haggle is a first project of its kind aiming to bring a content-centric solution for opportunistic networks comprised of mobile handheld devices [15]. Haggle presents a modular framework which can be extended to support a variety of applications and services embracing the ad hoc networking paradigm. At the core of the framework stands the search-based module which transparently matches data received during opportunistic communications servicing them to the application layer. Search is hence identified as a first-class operation, relying upon lower layer functionalities to achieve its application-specific goal.

ICEMAN follows in these steps even though it was designed for use in tactical scenarios [16]. It is seen as an evolution of the Haggle architecture with a rich set of capabilities aiming to provide reliable communication in sensitive and disruptive environments. This is achieved by introducing networking coding techniques and utility-based content caching at the transmission layer counteracting the disruptive nature of dynamic environments.

While the above projects are an interesting next step for AirCache, our aim is to analyze and study the underlying primitives needed to sustain the concept of floating data. Indeed, AC can be deployed to support a vast variety of crowdsensing applications, ranging from infotainment to urban security. For instance, an urban security application whereby users announce information pertaining to this context could be sustained by a nearby deployed AirCache. Information related to the context might involve a warning about a nearby hole which is announced and sustained within the AC and of interest to a blind person passing through. Free-speech manifestations might employ AirCache given its inherent benefits of being decentralized and operator-free. On the other hand, infotainment information could involve the dissemination of local events and goods which might expire later on. To this end, in the following section we review state-of-the-art work in the context of mobile networks similar in spirit to ours. Next, we delve into the technical details of our solution.

3. Related Work

We argued that data lifetime and accessibility in the AoI are ingredients of paramount importance which demand attention in order to enable our envisioned scenario. Once data is created or injected into the AC, a mechanism guaranteeing its immediate availability needs to be in place. A trivial approach whereby data is replicated upon each opportunistic encounter might seem as a solution to both requirements. However, this epidemic dissemination of data is paid in terms of device energy consumption and network storage as a whole.

To tackle the issues, we reviewed literature material in the context of content placement and replication techniques to check if prior work could serve our purposes. Such techniques have been vastly investigated in the realm of infrastructure networks where topological information is stable in time and is exploited to position data at fixed storage locations [17]. In our scenario, network topology is volatile and transient in time; hence these techniques cannot be adopted as they are.

Networking techniques relying on the assumption of a stable and global knowledge of network topology are unfeasible for generic OppNets. Tackling these issues in mobile, dynamic environments are HybridCache in [18] and Hamlet in [19] which rely on local information exchange among neighboring nodes to fulfill their respective objectives. Both approaches rely on local knowledge employing opportunistic caching techniques. Nodes in HybridCache achieve this by caching either the relayed data or the entire path toward that specific data content. The strategy employed depends on the size of the data being relayed which is a system parameter configured at system bootstrap.

Hamlet builds on this idea bringing it a step further. The caching strategy aims to create content diversity in the local vicinity and this is achieved by estimating the cached items in the neighborhood. Differently from HybridCache, where each node is an autonomous entity making decisions, in Hamlet there is a distributed, collective neighborhood effort trying to optimize resource allocation. Both these techniques rely on local information to perform their duty and, hence, lead to suboptimal, best-effort solution in achieving their goal.

Instead of binding to physical nodes, we can anchor (bind) data to geographical location. This way we are decoupling the content placement problem from the volatile network topology. In this direction, the authors of [20] propose a strategy whereby data is anchored at geographical locations and content replication degree inside an area is based on content popularity. The authors show through simulations that their proposal does outperform prior techniques in terms of data access costs. The proposals idea is to define multilevel grids covering a physical place, allocating popular contents to fine-grained grids. In this way popular content is more likely to be available nearby when compared to less popular content.

In order for the above scheme to reach its full potential, content popularity needs to be known in advance (a closed-world assumption) in order to compute the content anchoring strategy. Data in our scenario is produced and maintained by the network nodes itself. Considering this, the proposal above is reduced to simply maintaining an arbitrary number of replicas for each data in a bounded geographical area. Moreover, in our scenario data is produced locally and is volatile. If no mechanism guaranteeing data survivability (availability) is in place the data vanish.

Proposals [21, 22] address a scenario similar to ours. Through a fluid model, the authors in [21] derive a sufficient condition for the content to float with very high probability. The model takes into consideration the average number of nodes, the average node contact rate, and the average sojourn time of nodes in the AoI. The authors validate their model against simulation analysis showing that the derived condition is actually a necessary condition for the content to float with a high probability. To keep the model tractable, the authors make unrealistic assumptions regarding, for example, the dissemination strategy, assuming an epidemic approach to data dissemination. Following these steps, the authors in [23] provide evidence from a field trial employing a proof-of-concept implementation of a floating data application.

In the following, we introduce and discuss AirCache, a decentralized algorithm whereby nodes cooperate toward a common goal in order to guarantee data survivability inside an AoI. Our solution lends itself to another interesting feature of controlling the spatial distribution of data through the network, hence the possibility to control the data access costs. AirCache has a conservative approach on the data replication, taking into consideration energy and memory consumption of nodes.

4. AirCache: A Floating Data Network

We aim to devise a decentralized solution for the data anchoring problem whereby the data are binded to a geographical location of interest. The data are either produced locally or brought from elsewhere and need to be kept alive until some conditions are met. Indeed, the data might expire, for example, due to a time attribute attached to it expiring or as a consequence of node mobility and/or the absence of the necessary critical mass capable of sustaining an AC. From a practical point of view, a node possessing the data (e.g., the source node itself or a node which has taken this responsibility) performs a search for a nearby existing AC to which it can hook the data. If an AC is not present in the surroundings, the node takes the responsibility to create a new one. Independent of the starting conditions, once an AC is available, it is crucial to start replicating the data so as to have it persisting in the area regardless of node swarming.

Our solutions are designed to guarantee that a minimum number (Min) of replicas are always present in the system.

Of course, these criteria can only be enforced if the number of nodes required to fulfill is present in the anchoring zone. On the other hand, avoiding the negative effects of an epidemic dissemination, a good design choice is to also enforce an upper threshold (Max) of replicas. In specifics, the system allows a number of average (Avg) replicas in the AoI and exploits hysteresis in order to decide whether the replicas have to be augmented (Avg < Min) or diminished (Avg ≥ Max) so as to reach a stable situation where the number of replicas equals Avg. In this way, our system can provide guarantees that a balance of replicas between an upper and a lower number is always present.

Nodes in the system periodically broadcast their own capabilities translated in terms of energy, mobility, and memory space. This information is later on exploited whenever a content replication event is triggered. In these settings, a distributed algorithm establishing the grounds for node cooperation needs to be put in place. To this end, we delved into scientific literature and found an interesting solution that could suit our purposes. The Reliable R-Aloha (RR-ALOHA [24]) protocol is a distributed collision-free protocol employing a reservation mechanism, establishing the grounds for node cooperation. Through the rest of this section we detail our proposal built on top of the RR-Aloha protocol, discussing the added features and overall system *modus operandi*.

4.1. Joining the AC and Data Replication Policy. Reliable R-Aloha is a distributed MAC layer protocol employing a reservation mechanism for assigning dedicated communication slots to nodes in a distributed fashion. The reservation mechanism employs a *Frame Information* (FI) structure comprised of a certain number of *Basic Channels* (BCH) or communication slots of a certain duration which are contended by nodes. The protocol is collision-free and has been shown to solve the *hidden terminal problem*, providing the means for a reliable single-hop broadcast channel among neighboring nodes. In essence, RR-Aloha builds a slotted shared channel guaranteed to be accessed in mutual exclusion among nodes. Though the protocol is particularly apt to the slotted physical layers it could be ported on top of any physical layer, in particular on top of the 802.11 stack.

A node upon entering an AoI, a geographical location where some data needs to be anchored, goes into listening mode. While in listening mode, the node verifies whether there is an existing local AC nearby. If no local FI announcements are heard during this period, the node can proceed with creating a fresh AC; otherwise it enters a contention period (join-mode). In both cases, a node listens to nearby transmissions for a Frame Information Time (FIT) in order to reduce the probability of nearby collisions in case an AC is present. The current implementation considers a fixed FI length with a default value set to 100 BCHs. However, this is a system configuration parameter which can be set at system bootstrap. The consequence of a fixed FI length poses an upper bound on the number of nodes that can contend the slots in a FI. Also, as in the original design we further limit the number N of contending nodes for a FI to 50, leaving the rest of the communication slots for use in cases when additional bandwidth is required (later on).

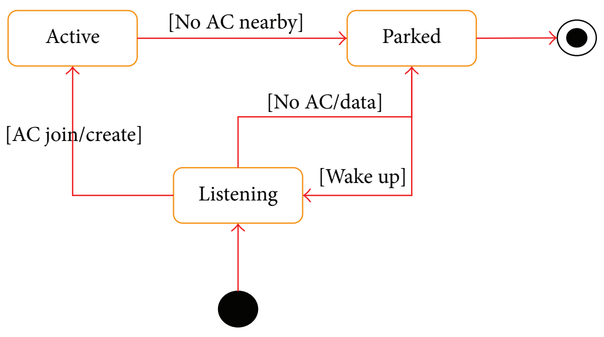

FIGURE 1: Diagram representing node state transitions.

After a FIT listening period, if the node has not sensed the presence of an AC and has data that require anchoring, it proceeds with the creation of the AC; otherwise it enters a contention period. The contention resolution period, which we discuss later in its fullness, ends with the node acquiring its own communication slot referred to as basic channel. Once the BCH is acquired, the node becomes active and participates in the system by broadcasting its view of the surroundings along with its chosen basic channel identifier appended at the *end* of the FI. Otherwise, if a node does not have any data content that needs anchoring and no local AC is sensed nearby, it is considered as parked. Parked nodes are not considered as part of the system but are eligible candidates at a later moment. Also, nodes can enter a parking mode whenever the AC has exhausted the available communication slots. Figure 1 depicts this entire process by showing the nodes state transition diagram.

Having introduced the basics of the AC mechanism, we need to discuss how the system enforces the replication policy. Each node, after Frame Information Time has gone by, has all the necessary information about the data replication level within the neighborhood or cluster (refer to Section 4.2). If the number of replicas for a particular data content is less than a threshold (Avg ≤ Min) and the system has the capability to further replicate it, a replication event is triggered. Contemplating for resource-consumption, the most capable nodes are involved in the replication process. The node eligible to broadcast a replica of the data is chosen based on the battery index (higher energy index) whereas the receiver(s) election is based on the buffer occupancy index (lower buffer occupancy). If the triggered event did not have any effect on the number of replicas the process repeats.

Similarly, if the maximum threshold is exceeded (Avg ≥ Max), the redundant replicas are discarded. The nodes eligible in this case are those with the lowest battery and buffer occupancy index and the resulting index is computed giving equal weight to both indexes.

4.2. Slot Reservation Mechanism. Before discussing the salient features of the AC protocol and delving into its algorithmic details, we start by providing a formal definition of cluster coined in the original RR-Aloha proposal.

Definition. Consider Cluster(C) = $\{\text{Node}(A) \wedge \text{Node}(B) \mid \text{Node}(A) \neq \text{Node}(B) \wedge \text{Node}(A) \in C \wedge \text{Node}(B) \in C \Leftrightarrow \text{Node}(A)$ hears $\text{Node}(B) \wedge \text{Node}(B)$ hears $\text{Node}(A)\}$.

That is to say, nodes participating within a cluster are nodes that can all hear and be heard by each other. In other words, a cluster is a strongly connected component of the temporal connection graph with vertexes where the nodes and the edges are bidirectional connections among these nodes.

A basic channel in the FI is either reserved or free. Nodes in the system periodically, after a FIT, confirm their reservation by broadcasting their perception of the surrounding neighborhood. Upon receiving this FI transmission, each node checks if its position as overheard by others is consistent with its view. If this is not the case, the node enters in contention mode for another slot. RR-Aloha adopts a sliding frame mechanism in order to compute this outcome whereby each node verifies the status of the BCH comprising the FI based on previous transmitted information from its single-hop neighboring nodes. We adopt a slightly different approach in processing the FI slot status occupancy as shown in Algorithm 1 and exemplified in Figure 2.

The original algorithm accounts only for a partial information of the overheard FIs, those parts of information falling inside the sliding frame of N slots. Differently from the original proposal, we exploit a node's local view in its fullness. The idea behind this design choice is due to the different mobility dynamics of vehicular network, context for which RR-Aloha was initially conceived, and human comprised networks. The former being more dynamic, demanding a more reactive algorithm for slot status computation. Our approach allows for a more complete picture of the slot reservation status, information which is not propagated but is instead used locally to reserve additional bandwidth whenever needed (later on). In this way, we are considering information which is at most $2 * \text{FIT}$ old. However, taking into account the human mobility in most scenarios and the 802.11 transmission range, this amount of time is negligible. As a future work, we plan to evaluate our approach when combined with optimal forwarding algorithms proposed in the context of vehicular networks [25, 26].

To implement the discussed features, we have devised some application level data structures as evidenced in Figure 3. The Frame Information is a data structure comprised of basic channels (BCHs), denoting the slot occupancy status of the 1-hop neighborhood. The BCH on its turn is also a data structure comprised of the following fields:

(1) A node identifier represented by an integer value holding the node identifier who has reserved that particular BCH, otherwise −1.

(2) The slot occupancy status represented by a flag with a true value if the BCH is *Free* or false if it is *Reserved* by some node.

(3) Another flag variable which is used to denote if a BCH is eligible for use in case some extra bandwidth is required for a transmission.

The *ReservationBoard* used by Algorithm 1 has structure similar to the FI with the difference that its contents denote the slot status occupancy in a 2-hop coverage area. The reason for holding a 2-hop slot occupancy status in the ReservationBoard is so as to implement the extra bandwidth reservation mechanism whereby nodes reserve additional transmission slots whenever the data to be transmitted cannot not fit inside a single BCH. If conditions are met and a node requires extra transmissions slots, it unilaterally decides and reserves the additional slots after consulting its local reservation board. Its intention is then notified in the outgoing FI. The additional slot reservations expire as soon as they are not needed anymore and this control is contemplated through lines 23–26 of Algorithm 1. This information is sent in broadcast once and only by the node requiring the slots.

Referring to Figure 3, *TableInfo* is yet another data structure holding the device capabilities and a list of content identifiers cached by the node. The identifier in practice might consist of the resulting hash value of the raw data content. Nodes broadcast this information in their reserved BCH along with the FI status. The replication mechanism exploits these data to compute and enforce the replication policy.

4.3. Controlling the Spatial Distribution of the Data. Data accessibility inside the Area of Interest is another important feature we deem worth pursuing. Depending on the data entry point in the AoI, data may get confined and replicated only in limited portions of the AoI. This happens because nodes within the AC start enforcing the replication policy only after having heard about the existence of the data. While it might happen that the data gets replicated in the entire AC, for example, due to mobility dynamics, this event is accidental rather than intentional.

Hence, it is necessary to provide a mechanism which can effectively control the spatial distribution of the data. The rationale being that a homogeneous data distribution in the area reduces data access costs, easing data availability in the AoI. Cluster intersection nodes are the best candidates which can be exploited to fulfill this objective. Intuitively, they are in the position that exposes the data to new areas which once introduced can be replicated if the conditions are met. To this end, after a FIT, nodes autonomously check whether they are positioned in a cluster intersection. This step involves partitioning the set of confirmed slots in the outgoing FI into disjoint clusters as detailed by Algorithm 2.

After completing the steps involved in Algorithm 1, nodes autonomously verify their position in their neighborhood in order to find out whether they are positioned at cluster intersections. The neighborhood connectivity graph is built by exploiting the received FIs transmissions. Through Algorithm 2 each node checks the received FIs whether they have an entry with a slot assigned to him (line 1).

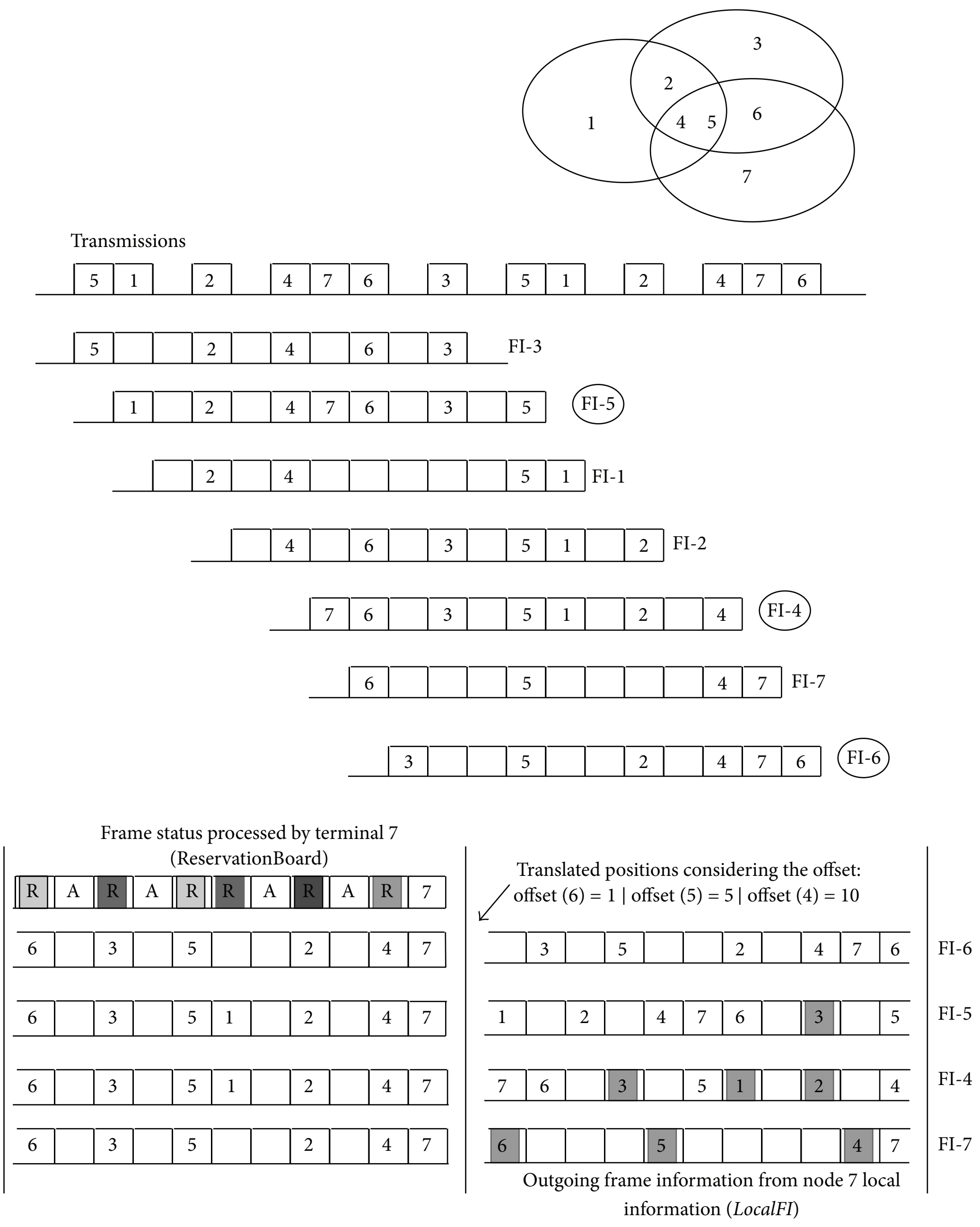
Transmissions
FI-3
FI-5
FI-1
FI-2
FI-4
FI-7
FI-6
Frame status processed by terminal 7 (ReservationBoard)
Translated positions considering the offset: offset (6) = 1 | offset (5) = 5 | offset (4) = 10
Outgoing frame information from node 7 local information (LocalFI)

Figure 2: Algorithm employed for slot status computation. In the upper part the FI composition transmitted by each node is shown while the bottom part exemplifies the steps involved in the computation of the ReservationBoard and of the outgoing FI from node 7 perspective. Circled nodes are part of node's 7 cluster.

This means that a bidirectional connection exists between the two. After this filtering process, the algorithm starts to partition the set of remaining nodes into disjoint clusters by exploiting the cluster property defined above (line 8). To avoid ambiguities, we point out that the computed clusters are not necessarily complete or entirely disjoint. The algorithm intentionally does not account for common elements (line 11) and incomplete information comes from the fact that we use only local information rather than a global view of the network. The purpose for this is related to the replication policy enforced by nodes found in cluster intersections which we detail below.

After nodes have inferred their respective positions in the neighborhood according to Algorithm 2, different behaviors are assumed depending on their role. Nodes present in at most one cluster react as explained through Section 4.1, while nodes found at cluster intersections, that is, nodes that participate in more than one connected cluster, have the responsibility to enforce the replication policy on a per cluster basis. This is achieved by making use of a special data packet which we call a *pull* data packet. This packet is sent in broadcast whenever the replica level inside a cluster is under the threshold and the cluster node has not a cached replica of the data itself. The replication criteria also take

```
input: FrameInformation (LocalFI), ReservationBoard (ReservationBoard), Received
FrameInformation(s) (ReceivedFIs), Local Node Identifier (NodeId)
output: Updated ReservationBoard and FrameInformation
(1)  bool Collision ← False, Owned ← False;
(2)  int Position ← −1; BCH Channel ← ∅;
(3)  for $i$ ← 1 to Length(LocalFI) do
(4)      BCH Heard ← BasicChannel(LocalFI, $i$);
(5)      foreach fi in receivedFIs do
(6)          Position ← Shift(fi, $i$);
(7)          Channel ← BasicChannel(fi, position);
(8)          Collision ← False;
(9)          Owned ← Owner(Channel);
(10)         if Owned and NotSameId(Channel, Heard) then
(11)             Collision ← True;
(12)             break;
(13)         end
(14)     end
(15)     if Collision then
(16)         UpdateBoard(Reservationboard, $i$, Free);
(17)         UpdateFI(LocalFI, $i$, Free);
(18)         if NodeId = Id(Heard) then
(19)             Collision(True);
(20)             EnterContention();
(21)         end
(22)     else
(23)         if (AdditionalAndExpired(Channel)) then
(24)             UpdateBoard(board, $i$, Free);
(25)         else
(26)             UpdateBoard(board, $i$, Reserved);
(27)         end
(28)     end
(29) end
```

Algorithm 1: Slot status computation.

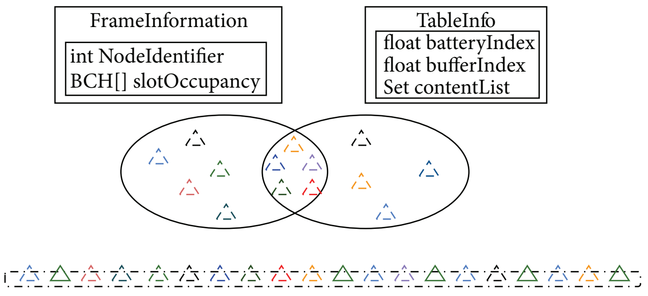

Figure 3: Devised data structures and respective data packets transmitted during the BCH. Green triangles denote free slots.

into account the number of nodes available in that particular cluster; that is, the minimum threshold is computed as the $\mathrm{Min}(\mathrm{Min}_{\mathrm{thresh}}, \mathrm{card}(C_i))$.

4.4. Discussion and Current Limitations. There are different design criteria and tradeoffs which need to be taken into careful consideration before deploying an AC. As discussed above, the FI has a fixed number of slots which needs to be decided *a priori*. Yet, a high number of slots means more nodes can participate in the system but at the same time it slows system reactivity to data replication. Symmetrically, a low number of slots could mean nodes being parked while having a reactive system which is paid in terms of battery consumption.

To tackle this issue, improvements have been proposed to the original RR-Aloha protocol. These include the possibility to dynamically adapt the Frame Information size by increasing or shortening its size depending on its level of fragmentation [27]. The fragmentation is measured by exploiting slots status information after a FIT. In our current implementation we do not contemplate this feature and leave it as a necessary future work.

A practical implication to the protocol's *modus operandi* is energy consumption due to the need to periodically reserve the communication slot. When considering an AC sustained by handheld portable devices this is indeed an issue which needs to be addressed more thoroughly. A way forward in addressing this issue might be to alternate nodes from active to parked mode and vice versa when certain criteria are met (e.g., the replica level is within the predefined thresholds).

```
     input: FrameInformation (LocalFI), Set (ReceivedFIs)
            Set (ConnectedComponents)
     output: Set of connected components ConnectedComponents
(1)  Set Connected ← Bidirectional(LocalFI, ReceivedFIs);
(2)  Cluster P;
(3)  P ← P ∪ FirstElement(Connected);
(4)  ConnectedComponents ← ConnectedComponents ∪ P;
(5)  foreach Slot in Connected do
(6)    bool InsideAnyCluster ← False;
(7)    foreach C in ConnectedComponents do
(8)      if IsMember(NodeId(Slot), C, ReceivedFIs) then
(9)        P ← P ∪ NodeId(Slot);
(10)       InsideAnyCluster ← True;
(11)       break;
(12)     end
(13)   end
(14)   if not InsideAnyCluster then
(15)     Cluster P_New;
(16)     P_New ← P_New ∪ NodeId(Slot);
(17)     ConnectedComponents ← ConnectedComponents ∪ P_New;
(18)   end
(19) end
```

Algorithm 2: Cluster set computation.

Yet, another issue is the AC interference phenomenon which occurs when different ACs come within communication range of each other. This interference can come as a result of a potentially different time synchronization of nodes participating in the different ACs. The phenomenon occurs even when a node caching data from an AC-1 moves outside the reach area of AC-1 but is still inside the AoI and has valid data that need anchoring. As per current design, this will trigger the node to create an AC-2 opened to other nodes to join in. At some point in time, the two ACs might end up interfering with each other.

The interference phenomenon undermines a factual deployment of the AC; hence we provide some insight as to how it might be addressed. The solution to this phenomenon might be to add an identifier (e.g., a timestamp of creation) to each autonomous AC which is propagated within the FI. This way nodes hearing transmissions originating from a different AC become temporarily parked. Interference between nearby communications is solved by nodes entering a recontention period for slots in their respective ACs. In this way, after less than a FIT, respective nodes can identify the presence of the other AC. Once interfering nodes are parked, an AC merger procedure could be put in place to handle the gradual merger of conflicting ACs with one another. Alternatively, nodes might stay parked until a single AC is present in their surroundings.

We deem this issue an important one, and how this interacts with a dynamic frame adaptation algorithm deserves further investigation. The experimentation part of this scenario is orchestrated in such a way so as to avoid this problem from occurring (refer to Section 5).

5. Simulation Environment and Evaluation Strategy

We chose to implement and evaluate our proposal in the Network Simulator 3 (NS3 [28]), a state-of-the-art simulation environment equipped with the 802.11 Phy/MAC stack. Before we delve into the details of the experimentation scenario, we briefly discuss some components of the simulation environment under scrutiny relevant to our work. In Figure 4 the *Node* component is shown which is an abstraction for the basic computing device. Nodes have certain capabilities which can be configured and acted upon at run-time. In our experimentation each node is equipped with a wireless 802.11 g interface and we adopt a flat addressing scheme, assigning each node a unique layer-3 address. This is of course a simplification as in a real deployment scenario this assumption is not valid anymore. However, the dynamic address assignment problem in decentralized environments is still an open research issue [29]; Level-2, MAC addresses are used in practice.

(1) *Channel and NetDevice.* They represent an abstraction for the transmission medium (either wireless or wired) and the network device respectively. In our simulation we adopt the wireless channel specification proposed by [30] which is featured in NS3.

(2) *Mobility Model.* To model the targeted scenario each node is configured with a proper mobility model. Indeed, in order to demonstrate the feasibility of our solution, we adopt the Random Way Point (RWP) mobility model which despite its simplicity provides some insights into the AC's behavior. The mobility

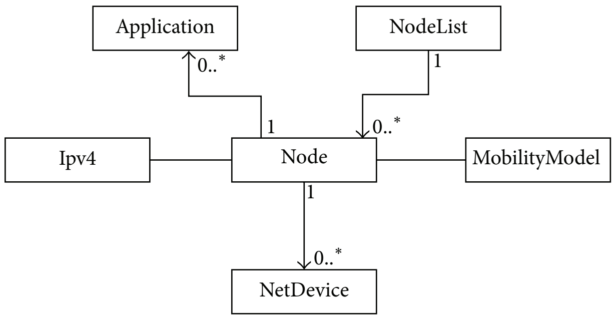

Figure 4: Node class diagram.

traces have been generated through the BonnMotion tool [31]. More details on this matter are given later on (refer to Section 6).

(3) *Application.* It represents the application layer where all the logic is implemented. The algorithms discussed throughout the paper have been implemented at this layer.

(4) *Energy Model.* It is yet another feature which can be attached to a node, accounting for the energy consumption due to communication. Our solution exploits this information when electing the sender/receiver nodes in the replication process.

The main objective of our proposal was to provide some guarantees of data survivability in the AoI. Therefore, we focus on the *TTL* metric, that is, the elapsed time interval from data being brought into the AC and the time they cease to exist. Yet another mandatory requisite was the control of spatial data distribution within the AC. Intuitively, a more homogeneous distribution of the data could ease data availability in the AoI. To this end, demonstrating the feasibility of our approach, we perform different simulations with varying node density and mobility characteristics which are discussed through Section 6.

Recalling the argument stated in Section 4.4, our protocol in its current implementation does not contemplate for the scenario of conflicting ACs. In the simulations this situation is avoided by considering a single AoI, the same for all data contents present in our simulation environment. Nodes in the simulation scenario can only join and not create the AC, while the node creating (bootstrapping) the AC is chosen randomly at the beginning of the simulation time from the entire node population.

6. Results

Through this section we discuss the simulation strategy and outcome. The evaluation metrics we employ are the time interval the data survives within the AoI and a distance metric formulated as shown below:

$$\text{Distance}_{\text{data}} = \frac{1}{N} \times \sum_{n \in \text{AoI}} \text{dist}_{\min}(n, \text{datum}). \quad (1)$$

Table 1: Parameters for the RWP mobility model used for the evaluation.

Simulation area	400×400
Mobility	$[0.5, 1.5]$ m/s
Pause time	30 s
Trace duration	60 min
Attraction point	200×200
AoI radius	150 m

The metric measures the minimum number of hops required to reach the data, averaged over the population of nodes sustaining the AC. The set $n \in$ AoI includes all nodes inside the AoI even if they have not yet joined the AC. Nodes caching a replica of the data are not considered. As anticipated, the mobility model used to evaluate our proposal is the RWP mobility model generated with the parameters shown in Table 1 and a cut-off time of 3600 s.

To activate relevant features of our protocol, each node in the population is attributed an initial battery level taken uniformly at random from the range [500, 1750] mAh. We do not account for the buffer occupancy level and the sender/receiver of data is chosen by exploiting the battery level only. Each studied scenario we perform 20 runs per configuration employing different mobility seeds. As discussed in Section 5, we avoid the AC interference phenomenon by delegating the responsibility of AC creation to a single node chosen at random node within the AoI and restrict the others to only join it. Nodes might still move away from the original AC, giving rise to different ACs which are time-synchronized with one another. The AC merger process in this scenario is left to the protocol without additional intervention; conflicting nodes (if any) enter a recontention period for new available slots.

6.1. Data Survivability in the AoI. Our first metric of relevance is data survivability within the AoI. To this end, we generate different contact traces with different seeds employing the parameters evidenced in Table 1. The AoI is a circle within the simulation world and nodes possessing a replica of the data once outside the AoI discard the data along with the AC related information and are marked as inactive. We study the system behavior by imposing some control over the in/outflow of nodes leaving and entering the AoI.

To this end, the nodes that fall outside the AoI before simulation starts are marked as inactive. Inactive nodes become active when entering the AoI and the number of activated ones depends on the flow of nodes that have left the AC. In order to enforce this in/out policy within the AoI, we take a snapshot of the simulation world every 5 s and control the inflow depending on the outflow of the last snapshot.

We study the data survivability with varying population size and different in/out policies. Each scenario is simulated 20 times employing different mobility seeds in order to increase the confidence of the results. The considered data size along with the control data fits inside a single BCH; hence there is no need for extra slot reservation. The results are shown in Table 2.

TABLE 2: Average AC survival times in minutes with varying population size and different in/outflow policy.

	Trace	1 : 1	1 : 2	1 : 3	1 : 4	1 : 5	Departure
			Population size 10				
Average	48.53	39.05	22.11	20.11	10.1	7.24	6.2
Median	42.12	30.55	18.42	17.23	6.56	4.54	4.3
			Population size 15				
Average	52.12	49.11	40.2	30.23	25.23	20.53	6.3
Median	43.5	40.57	33.47	23.47	18.34	15.45	4.5
			Population size 20				
Average	55.21	51.56	48.45	33.45	27.57	25.16	6.55
Median	53.32	49.06	46.45	32.29	25.54	23.46	5.5
			Population size 25				
Average	60	59.51	54.11	50.01	45.55	38.51	7.02
Median	60	56.53	50.37	44.4	39.5	30.36	4.5
			Population size 30				
Average	60	60	58.11	55.12	51.34	48.54	6.76
Median	60	60	55.45	50.7	49.23	40.32	7.3
			Population size 35				
Average	60	60	60	60	58.12	49.11	7.54
Median	60	60	60	60	52.18	44.5	8.12

The scenario denoted with *Trace* serves as the benchmark solution where no in/outflow policy is applied; that is, the mobility dynamics inside the AoI are those exhibited by the original trace. The last column (*Departure*) denotes the average time interval that the producer nodes left the AoI for each configuration run.

As it is shown by the outcome, the AC is able to provide a margin of profit which decreases when the control flow policy becomes more aggressive. Overall the obtained results evidence the importance of a mechanism guaranteeing data availability in the AoI and at the same time show that AC serves its purpose. An increase in the node population size corresponds to an increase of data lifetime. Similarly, when a 1 : 6 policy is enforced the protocol is not as capable at counteracting the effects of nodes departing the AoI. This trend is exhibited in all the mobility traces. However, even in this case we are able to provide a margin of profit when compared to a scenario where no cooperation is involved.

6.2. Controlled Data Distribution. A homogeneous data distribution eases data accessibility in the AoI. As discussed, this control is enforced by cluster intersection nodes whenever conditions are met. To measure the benefits of this feature we contrast the approach with the one where no distribution control is enforced. To this end, we have orchestrated a set of simulations, 10 for each configuration run, with the parameters evidenced in Table 3.

We point out that, in the scenario where no distribution control applies, the data does not survive the end of the trace in 3 different runs with a data lifetime of 2.45, 5.45, and 6.34 min. We do not consider this particular runs when averaging the distance metric reported in Figure 5. An important effect of applying distribution control is resiliency to mobility when compared to the strategy where no control is applied. In the controlled approach the data are replicated more rapidly, creating a distributed number of replicas over the AoI rendering the AC data less subject to mobility of nodes in a certain areas. The distance metric for the distribution control scenario as in the graph converges to 1 more rapidly. In the case where no control is applied, converging to 1 requires more time.

TABLE 3: Settings of the controlled distribution scenario.

Simulation area	500 × 500
Mobility	[0.5, 1.5] m/s
Pause time	30 s
Trace duration	10 min
Population size	35
Attraction point	250 × 250
AoI radius	200 m

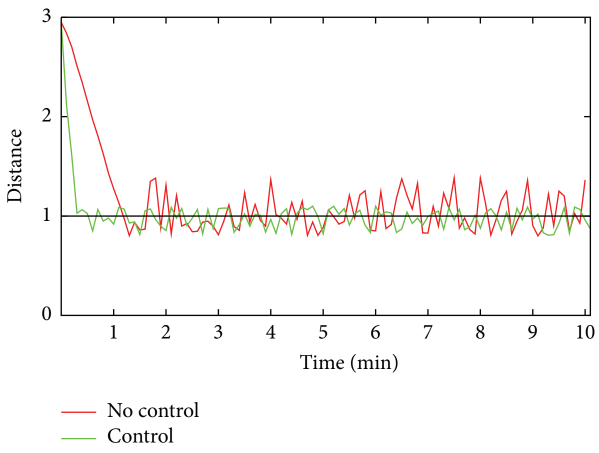

FIGURE 5: Comparison of the strategy employing a controlled distribution approach versus the default one. A Min/Max policy of 1 : 2 is employed.

The process of data replication in the no control scenario is incidental rather than intentional. Indeed, the scenarios where data is replicated in other places that differ from the initial point of entry depend on the mobility dynamics and/or the chance of cluster nodes caching the data. Also, when the data has reached a stable replica level within the AC, the no control scenario is subject to more fluctuations while the controlled scenario exhibits a more stable trend. This fluctuation difference between the two schemes stands in their *modus operandi* where in the control scenario data the replication policy is enforced in a more timely fashion while in the no control policy it depends upon node mobility physically carrying the data to newly arrived nodes.

7. Conclusion

In this paper we have presented AirCache, a solution designed to enable a floating data network in a distributed and collision-free way. In the considered scenario, users voluntarily provide data, which is then maintained in an AoI through replication among passing-by or stationary users. In our experiments, we have considered an AoI composed by mobile wireless nodes and study the system behavior under different scenarios.

Our tests have shown that AirCache can be effectively deployed to guarantee data availability in the AoI. In other words, we can push data storage and consumption close to the network edges instead of congesting Internet links even in presence of a *local scopus*. Clearly, node density in the considered AoI is a crucial factor to ensure the survivability of the system, thus requiring AirCache to be employed in areas where people tend to gather.

AirCache also includes a node election strategy to improve the global energy/memory consumption needed to maintain the data floating in the AoI. However, this strategy can be improved, for instance, by dynamically tuning the slot length depending on the mobility dynamics. We plan to add this feature as future work. Furthermore, we also plan to implement some real application on top of our system to test how it could support it; to this aim, an interesting case study would be represented by mobile games [32, 33].

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

Acknowledgments

This work was partially supported by the University of Padua through Projects CPDA151221 and CPDA137314.

References

[1] V. Cerf and M. Senges, “Taking the internet to the next physical level,” *IEEE Transactions on Mobile Computing*, vol. 49, no. 2, pp. 80–86, 2016.

[2] Cisco, *The Zettabyte Era—Trends and Analysis*, 2015.

[3] M. S. Desta, E. Hyytia, J. Ott, and J. Kangasharju, “Characterizing content sharing properties for mobile users in open city squares,” in *Proceedings of the 10th Annual Conference on Wireless On-Demand Network Systems and Services (WONS ’13)*, pp. 147–154, Banff, Canada, March 2013.

[4] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson, “People-centric urban sensing,” in *Proceedings of the ACM Annual International Workshop on Wireless Internet (WICON ’06)*, New York, NY, USA, 2006.

[5] G. Smith, R. Wieser, J. Goulding, and D. Barrack, “A refined limit on the predictability of human mobility,” in *Proceedings of the 12th IEEE International Conference on Pervasive Computing and Communications (PerCom ’14)*, pp. 88–94, Budapest, Hungary, March 2014.

[6] A. Bujari, *Opportunistic data gathering and dissemination in urban scenarios [Ph.D. dissertation]*, Alma Mater Studiorum Univesity of Bologna, Bologna, Italy, 2014.

[7] C. E. Palazzi, A. Bujari, G. Marfia, and M. Roccetti, “An overview of opportunistic ad hoc communication in urban scenarios,” in *Proceedings of the 13th Annual Mediterranean Ad Hoc Networking Workshop (MedHocNet ’14)*, pp. 146–149, Piran, Slovenia, June 2014.

[8] P. Ruiz and P. Bouvry, “Survey on broadcast algorithms for Mobile Ad Hoc Networks,” *ACM Computing Surveys*, vol. 48, no. 1, article no. 8, 2015.

[9] C. E. Palazzi, A. Bujari, and E. Cervi, “P2P file sharing on mobile phones: design and implementation of a prototype,” in *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT ’09)*, pp. 136–140, Beijing, China, August 2009.

[10] S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni, “DTN protocols for vehicular networks: an application oriented overview,” *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 868–887, 2015.

[11] L. Pelusi, A. Passarella, and M. Conti, “Opportunistic networking: data forwarding in disconnected mobile ad hoc networks,” *IEEE Communications Magazine*, vol. 44, no. 11, pp. 134–141, 2006.

[12] A. Bujari, C. E. Palazzi, D. Maggiorini, C. Quadri, and G. P. Rossi, “A solution for mobile DTN in a real urban scenario,” in *Proceedings of the IEEE Wireless Communications and Networking Conference Workshops (WCNCW ’12)*, pp. 344–349, Paris, France, April 2012.

[13] A. Bujari, S. Gaito, D. Maggiorini, C. E. Palazzi, C. Quadri, and G. P. Rossi, “Delay tolerant networking over the metropolitan public transportation,” *Mobile Information System*, In press.

[14] R. K. Ganti, F. Ye, and H. Lei, “Mobile crowdsensing: current state and future challenges,” *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[15] J. Scott, P. Hui, J. Crowcroft, and C. Diot, “Haggle: a networking architecture designed around mobile users,” in *Proceedings of the 3rd International Conference on Wireless on Demand Network Systems and Services (WONS ’06)*, pp. 78–86, Les Menuire, France, January 2006.

[16] S. Wood, J. Mathewson, J. Joy et al., “Iceman: a practical architecture for situational awareness at the network edge,” Tech. Rep., 2013.

[17] I. Baev, R. Rajaraman, and C. Swamy, “Approximation algorithms for data placement problems,” *SIAM Journal on Computing*, vol. 38, no. 4, pp. 1411–1429, 2008.

[18] 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.

[19] M. Fiore, F. Mininni, C. Casetti, and C.-F. Chiasserini, “To cache or not to cache?” in *Proceedings of the IEEE INFOCOM*, pp. 235–243, IEEE, Rio de Janeiro, Brazil, April 2009.

[20] S.-B. Lee, S. H. Y. Wong, K.-W. Lee, and S. Lu, “Content management in a mobile ad hoc network: beyond opportunistic strategy,” in *Proceedings of the IEEE INFOCOM*, pp. 266–270, Shanghai, China, April 2011.

[21] E. Hyytiä, J. Virtamo, P. Lassila, J. Kangasharju, and J. Ott, “When does content float? Characterizing availability of anchored information in opportunistic content sharing,” in *Proceedings of the (INFOCOM ’11)*, pp. 3137–3145, Shanghai, China, April 2011.

[22] J. Virtamo, E. Hyytiä, and P. Lassila, “Criticality condition for information floating with random walk of nodes,” *Performance Evaluation*, vol. 70, no. 2, pp. 114–123, 2013.

[23] S. Ali, G. Rizzo, V. Mancuso, V. Cozzolino, and M. A. Marsan, “Experimenting with floating content in an office setting,” *IEEE Communications Magazine*, vol. 52, no. 6, pp. 49–54, 2014.

[24] F. Borgonovo, A. Capone, M. Cesana, and L. Fratta, “Adhoc: a new, flexible and reliable MAC architecture for ad-hoc

networks," in *Proceedings of the IEEE Wireless Communications and Networking*, pp. 965–970, New Orleans, La, USA, 2003.

[25] G. Marfia, A. Amoroso, M. Roccetti, and G. Pau, "Safe driving in LA: report from the greatest intervehicular accident detection test ever," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 2, pp. 522–535, 2013.

[26] A. Amoroso, G. Marfia, and M. Roccetti, "Going realistic and optimal: a distributed multi-hop broadcast algorithm for vehicular safety," *Computer Networks*, vol. 55, no. 10, pp. 965–970, 2011.

[27] J. Liu, F. Ren, L. Miao, and C. Lin, "A-adhoc: an adaptive real-time distributed MAC protocol for vehicular Ad Hoc networks," *Mobile Networks and Applications*, vol. 16, no. 5, pp. 576–585, 2011.

[28] https://www.nsnam.org/documentation/.

[29] M. Fazio, C. E. Palazzi, S. Das, and M. Gerla, "Facilitating real-time applications in VANETs through fast address auto-configuration," in *Proceedings of the 4th Annual IEEE Consumer Communications and Networking Conference (CCNC '07)*, pp. 981–985, Las Vegas, Nev, USA, January 2007.

[30] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proceedings of the ACM Workshop on ns-2: the IP Network Simulator (WNS2 '06)*, New York, NY, USA, 2006.

[31] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "Bonnmotion: a mobility scenario generation and analysis tool," in *Proceedings of the Annual International Conference on Simulation Tools and Techniques (SIMUTools '10)*, pp. 1–10, Malaga, Spain, March 2010.

[32] M. Furini, "Mobile games: what to expect in the near future," in *Proceedings of the GAMEON Conference on Simulation and AI in Computer Games*, Bologna, Italy, 2007.

[33] M. Gerla, D. Maggiorini, C. E. Palazzi, and A. Bujari, "A survey on interactive games over mobile networks," *Wireless Communications and Mobile Computing*, vol. 13, no. 3, pp. 212–229, 2013.