CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2015; **27**:3295–3315
Published online 4 February 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3221

SPECIAL ISSUE PAPER

Combining geo-referencing and network coding for distributed large-scale information management†

Marco Picone[1], Michele Amoretti[2,*,†], Marco Martalò[1,3], Francesco Zanichelli[1] and Gianluigi Ferrari[1]

[1]*Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma, Parma, Italy*
[2]*Centro Interdipartimentale SITEIA.PARMA, Università degli Studi di Parma, Parma, Italy*
[3]*E-Campus University, Novedrate, Italy*

SUMMARY

The widespread and ubiquitous availability of Internet access enables the collective sharing of huge amount of data generated by heterogeneous sources. For example, the information, which will be exchanged among entities (sensors, people, and services) of future smart cities to enhance the security and lifestyle of their citizens, poses the challenging question of how this information can be efficiently and effectively maintained across the city. In this article, we propose a decentralized approach, based on the distributed geographic table (DGT) overlay scheme, which exploits geo-referenced information about nodes to achieve efficient data management. After recalling DGT main concepts, we illustrate the possible node types and how information can be published and retrieved within the network. To cope with the unavoidable node failures and disconnections, our approach leverages upon randomized network coding to increase the robustness of publish/retrieval operations. Evaluation is carried out through an extensive simulation analysis for a realistic urban scenario using the metrics of efficiency in data publication/search, resource availability, and storage occupancy requirements. Results show the approach effectiveness for large-scale sharing of geo-referenced information and tradeoffs between redundancy overhead and resource availability. A few results obtained with a preliminary DGT implementation are also presented in the paper. Copyright © 2014 John Wiley & Sons, Ltd.

Received 26 March 2013; Revised 14 October 2013; Accepted 23 December 2013

KEY WORDS: peer-to-peer (P2P); network coding (NC); smart cities

1. INTRODUCTION

The concept of smart city has recently emerged from the interaction of research areas like *intelligent cities* [1] and *smart communities* [2]. Cities can be considered as systems of systems and offer tremendous opportunities for the introduction of digital nervous systems, intelligent responsiveness, and optimization at every level of system integration. For instance, connected vehicles can participate in the smart mobility subsystem by sharing traffic and travel data as well as receiving information about the nearest available parking slot and other relevant events. In this context, one major aspect being discussed in the research community is the choice between centralized and decentralized organizations for infrastructures, which need to ensure robust and secure data storage

†This paper has been partially presented at the *International Conference on High Performance Computing & Simulation* (HPCS), Madrid, Spain, July 2012.

*Correspondence to: Michele Amoretti, Centro Interdipartimentale SITEIA.PARMA, Università degli Studi di Parma, Parma, Italy.

†E-mail: michele.amoretti@unipr.it

and retrieval [3–5]. The availability of massive amounts of sensed information provides fascinating opportunities to understand the city activity by means of modeling and analysis. A large amount of information (either raw or aggregated) generated by city actors (both humans and machines) needs to be stored, maintained, and returned, either in proactive or reactive manners, to city actors themselves.

In this article, we propose an efficient scheme for maintaining and distributing large amount of information. Our approach consists in defining a peer-to-peer (P2P) network, based on the distributed geographic table (DGT) overlay scheme [6, 20], which allows to store and distribute city status information with minimum overhead and high reliability. The use of network coding (NC), and in particular randomized network coding (RNC), allows the achievement of these goals [7]. Data regeneration is also necessary to improve the robustness of the system against possible losses. While the DGT approach obtains load balancing and avoids the presence of bottlenecks and single points of failure, the use of RNC techniques allows data redundancy, which makes information retrieval extremely reliable and real-time streaming highly efficient. Indeed, RNC improves the performance of the DGT, mitigating the block transfer scheduling or piece selection problem, especially when nodes dynamically join/depart from the Internet. Moreover, RNC is important also for another functionality of the system, that is, distributed storage: should a storage node fail, the stored information could be retrieved by properly combining the information contained in other storage nodes. Therefore, the system architecture is based on two distinct yet complementary pillars: RNC provides tunable data redundancy for high availability, whereas geo-referencing enables searching for data fragments in the most probable direction.

Although the idea of combining NC techniques and P2P architectures has already been investigated in the literature (see, e.g., [8, 9] and references therein), our work is novel in the sense that it exploits these ideas in a properly designed architecture, which includes different types of peers—data sources, storage nodes, data aggregators, and user nodes—that may be stationary or mobile. The DGT overlay scheme, which takes into account the geographic positions of the peers [20]. The DGT efficiently supports the publication and search of geo-localized data and, interacting with RNC functionalities, allows to maintain the published data either periodically or sporadically. A Sip2Peer-based implementation of the proposed system is under development, with application to a realistic smart city scenario imagined for the city of Parma, Italy.

The paper is organized as follows. In Section 2, we briefly discuss the reference works in the fields of P2P and NC. In Section 3, our joint DGT/RNC-based architecture for information maintenance and distribution is illustrated. In Section 4, we derive a simple, but insightful, analytical framework to characterize the performance, in terms of publication and search hops, of the proposed architecture. In Section 5, the performance of the proposed solution is evaluated, through simulations, considering realistic dynamic scenarios and validating the analytical results. A preliminary DGT implementation, based on the open source Sip2Peer middleware, and corresponding experimental results are presented in Section 6. Finally, Section 7 concludes the paper.

2. RELATED WORK

The adoption of fully decentralized approaches for highly pervasive monitoring, data aggregation, and information sharing within cities is a new research topic that is gaining momentum because of the availability of a new generation of smart devices. In particular, the P2P paradigm enables two or more entities to collaborate spontaneously in a network of 'equals' (peers) by using appropriate information and communication systems without the need for central coordination. In the last decade, P2P has been studied and applied to different application scenarios, from file sharing to live multimedia streaming [10]. Recent research is focusing on P2P-based large-scale storage systems [11], distributed hash tables [12], social networks [13], and measurements of real systems [14]. Particularly promising are the P2P approaches based on geographic localization, for example, Globase.KOM [15] and those based on traffic information [4].

NC is a recently proposed network-oriented channel coding paradigm, arisen in the field of information theory, which generalizes the classical concept of routing in wired networks [16]. With NC, in fact, intermediate nodes are not only allowed to forward incoming packets but also to encode

DOI: 10.1002/cpe

them. This allows to achieve the multicast capacity [16] and, therefore, leads to potential advantages in terms of bandwidth and computational efficiency, robustness, and other performance metrics. Although NC has been extensively studied from a theoretical point of view, several practical scenarios, where benefits can be observed, have been proposed in the last years. An example of application domain is distributed storage [7]. In a previous work, we illustrated how NC and P2P can be enabling technologies for robust distributed storage [17]. NC has also been considered in [18], where the authors have implemented a prototype NC/P2P filecasting system and tested it in the distribution of large files (e.g., several GBytes) over the Internet. Their experimental results are very encouraging, although the considered scenario is quite simpler than the one we address in this paper.

3. SYSTEM ARCHITECTURE

In this section, we illustrate our distributed architecture for the management of information flows in smart cities. We first recall the main features of the DGT overlay scheme, which allows every node to maintain knowledge about surrounding peers, as well as to publish and retrieve data items. We also illustrate how NC techniques are exploited to ensure data survival. Note that these two parts currently focus on different system aspects: RNC allows to have tunable data redundancy for high availability, whereas geo-referencing enables searching for data fragments in the most probable direction. Future work will investigate the possibility of tighter interoperation between the two pillars of system architecture.

3.1. Distributed geographic table

A *structured* decentralized P2P overlay is characterized by a controlled overlay, shaped in a way that resources (or resource advertisements) are placed at appropriate locations [10]. Moreover, a globally consistent protocol ensures that any node can efficiently route a search to some peer that has the desired resource, even if the resource is extremely rare. Beyond basic routing correctness, two important constraints on the topology are (i) a sufficiently small maximum number of hops in any route, so that requests complete quickly and (ii) a sufficiently small maximum number of neighbors of any node, so that maintenance overhead is limited.

The DGT is a structured overlay scheme where each participant can efficiently retrieve node or resource information (data or services) located near any chosen geographic position [19, 20]. In such a system, the responsibility for maintaining information about the position of active peers is distributed among nodes, for which a change in the set of participants causes a minimal amount of disruption. The interested reader on the main DGT concepts is referred to the previous work by Picone *et al.* [20]. In the following, we focus on the routing strategy, whose proper extension to the smart city scenario of interest is one of the novel contributions of this work. DGT routing parameters are summarized in Table I and explained in the following.

The DGT routing strategy is a function R that, given a geographic position ω, returns the set of nodes $N(\omega, a)$ that are in the interest region a and satisfy the request. Therefore, a routing query issued by node p has the following structure: $route(p, \omega, a)$. The routing strategy is used not only to maintain the neighborhood of a peer but also to discover active peers in a remote region of interest.

Table I. DGT routing parameters.

ω	Geographic position
a	Interest region
$N(\omega, a)$	Nodes in region a centered in ω
R	Routing function
p	Generic node
$route(p, \omega, a)$	Routing query
Δt	Time range of a query's reply

DGT, distributed geographic table.

DOI: 10.1002/cpe

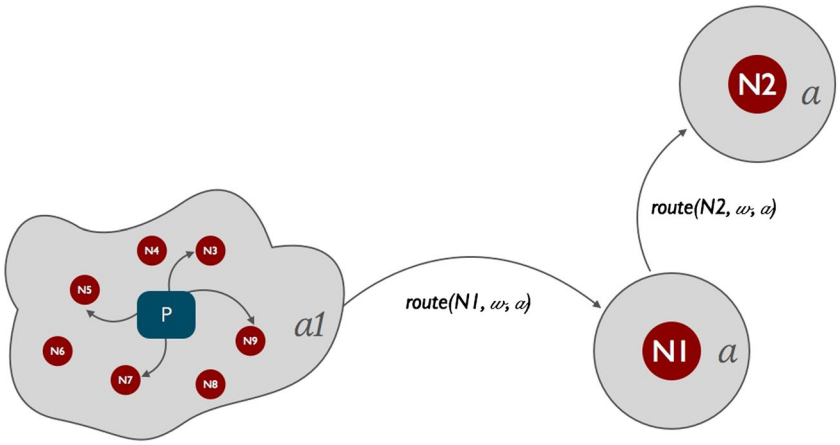

Figure 1. Propagation of a query between nodes to retrieve the neighborhood of a remote region of interest.

Figure 1 shows the propagation of a query between nodes, to retrieve the neighborhood of a remote region of interest.

The query propagation process is affected by (i) the distance between the source and the destination of the query itself and (ii) the size of the neighborhood region. Two kind of queries are envisioned. The first query returns the list of nodes that are interested on a type of data within a region centered in a specific position, defined by its latitude and longitude—such a list can be used for publication purposes. The second query returns the list of nodes that own a type of data, in an area that is centered in a specific position defined by its world's coordinate, being such data not older than the specified time range Δt. Such a list can be used for retrieval purposes.

3.2. Node description

The envisioned application of interest is associated with the production and consumption of information relevant to citizens about the status of smart cities. Such an information is typically generated by monitoring subsystems and is usually maintained and disseminated by some networking subsystems. To this purpose, four types of peers are envisioned in the proposed architecture.

A *raw data source (RDS)* generates basic data pieces, does not usually have local storage capabilities, and it is typically associated to specific sensors monitoring critical aspects (e.g., traffic cameras, pollution level sensors, etc.). A *storage node (SN)* is able to store the (possible large) amounts of data produced by monitoring subsystems. An *aggregated data source (ADS)* is not only a consumer of data produced by RDS but also a producer of filtered/aggregated data representative of a concise 'picture' of the city status at a given place and/or time. A *user node (UN)* is interested in receiving and visualizing raw or aggregated data, related to any point of interest, in real time. Figure 2 shows a set of interacting nodes and highlights the proposed architecture, which is composed by common layers (network, communication middleware, and DGT), supporting the specialized functionalities of the different node types.

Each peer has a PeerDescriptor, which contains the following information: (1) node type identifier (i.e., RDS, ADS, UN, SN); (2) communication reference (i.e., source/destination IP, port, and proxy); (3) prioritized list of locations of interest and data types that the nodes want to proactively receive; and (4) list of generated data types. In particular, the last two items are envisioned, in our architecture, to inform other nodes about 'who has what' and 'who needs what'. This allows the design of efficient information distribution/retrieval algorithms. Each information element produced by a peer has a DataDescriptor, which contains a key, the geographical coordinates of the location in which it has been generated, the data type, and the time validity of the data item. If the data item is a fragment of a larger data item, the DataDescriptor contains also the ordering number. Each information item is finite, has a precise position in space and time, and is uniquely identifiable by a key, which is generated by hashing the item descriptor.

DOI: 10.1002/cpe

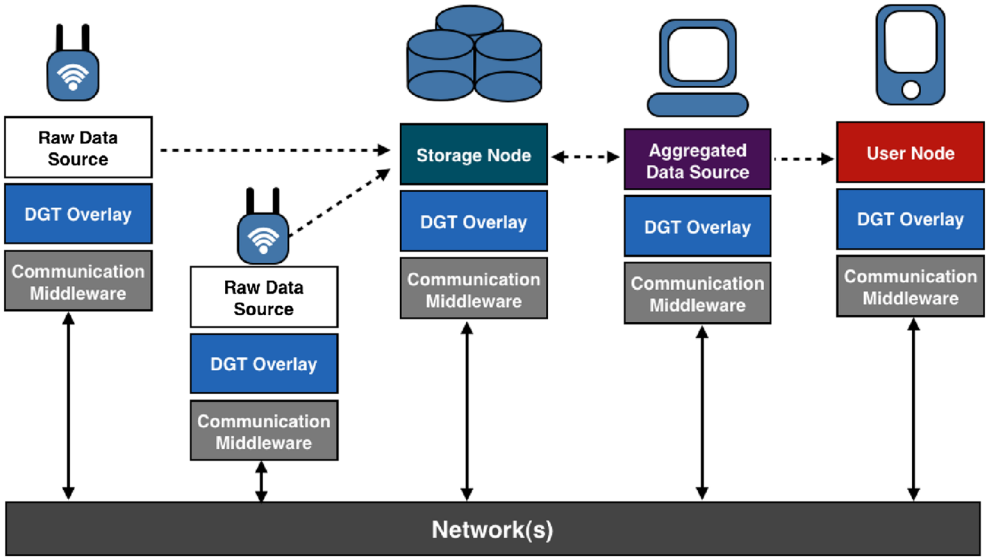

Figure 2. The proposed architecture illustrated by means of interacting nodes (with different specializations).

Table II. User mobility parameters.

ℓ	Generic path
$v_{\text{des}}^{(\ell)}$	Desired speed of a vehicle (dimension: [km/h])
v_{min}	Minimum vehicle speed (dimension: [km/h])
$v_{\text{max}}^{(\ell)}$	Speed limit related to the path (dimension: [km/h])
δ_{jam}	Critical vehicular density for which a traffic jam appears (dimension: [veh/km])
δ	Linear density of vehicles along the roads (dimension: [veh/km])

In general, all node types can support mobility. However, in this paper we assume that all nodes are static, with the exception of UNs associated with vehicles, whose street mobility is modeled according to the fluid traffic model (FTM) [21]. The FTM accurately describes, without high computational requirements, urban scenarios characterized by different speed limits for the various virtual paths. This model is based on realistic relationships between relevant traffic parameters (namely, speed, density, and volume) and was developed for cellular systems, both in ideal conditions and with reference to real street maps. The FTM describes speed as a monotonically decreasing function of vehicular density, forcing lower speeds when the traffic congestion reaches a critical point. In particular, the desired speed of a vehicle moving along the ℓ-th possible path is given by

$$v_{\text{des}}^{(\ell)} = \max\left\{v_{\text{min}}, v_{\text{max}}^{(\ell)}\left(1 - \frac{\delta}{\delta_{\text{jam}}}\right)\right\} \tag{1}$$

where $v_{\text{des}}^{(\ell)}$ is the evaluated desired speed (dimension: [km/h]), v_{min} is the minimum vehicle speed according to its structural characteristics (dimension: [km/h]), $v_{\text{max}}^{(\ell)}$ is the speed limit related to the path (dimension: [km/h]), δ_{jam} is the critical linear vehicular spatial density (dimension: [veh/km]) for which a traffic jam appears, and δ is the current linear vehicular spatial density (dimension: [veh/km]) along the road. All these parameters are listed in Table II. The rationale behind (1) is that the higher the node density, the lower the user speed should be, because nodes cannot move at the highest possible speed. Therefore, (1) takes into account a linear speed reduction due to an increase in the street traffic. Moreover, if the node density reaches a critical limit for which a traffic jam appears, the model allows nodes to move from that density to the lowest possible speed because of the presence of a traffic jam itself.

DOI: 10.1002/cpe

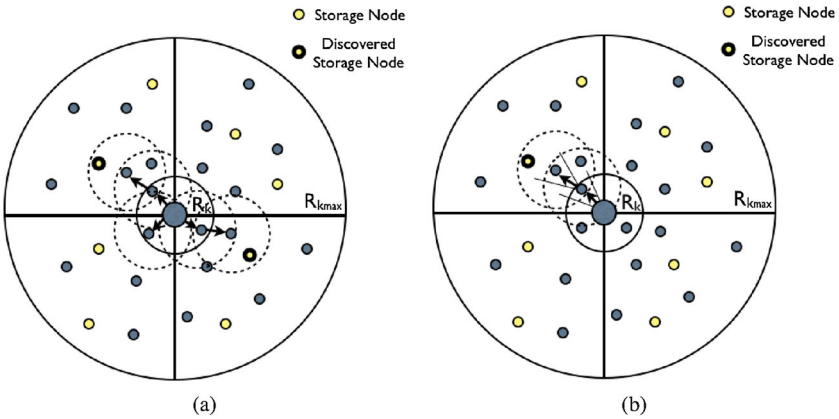

Figure 3. In the publication process (left image), messages are propagated in the four quarters of the circular region centered in the node. On the other hand, in the search process (right image), messages are propagated only in the quarter of the target location.

3.3. Publication process

The process for publishing a data item is carried out as follows. If the publisher knows **less than** N_s **SNs**, it does create publication request messages, having the structure described as in Subsection 3.1, which are propagated in a circular region of radius $R_{k_{\max}}$ (dimension: [km]), as shown in Figure 3 (left). More precisely, each message is sent to the neighbors in the first geobucket, which in turn may propagate it, skipping the nodes that have already received it. Message forwarding is limited to the nodes, which are within the circular region of radius $R_{k_{\max}}$, centered in the request originator. As we show in Section 4, the resulting number of propagation hops for each request message, until N_s SNs are discovered, is usually reasonably limited. As a result of the message propagation process, the publisher obtains a list of SNs that are interested in maintaining the data item.

From each data item to be published, a set of fragments is generated, using an RNC strategy, which will be detailed in Subsection 3.5. Such fragments (identified by unique keys and by the key of the data item they have been generated from) are uniformly distributed among the SNs that belong to the list obtained in the previous phase. In particular, the following algorithm is applied. The area where the fragments have to be published is divided as shown in Figure 3. A fraction of the fragments is published in the small circular region with radius R_k (dimension: [km]). The other fragments are published in the circular crown whose internal and external radiuses are R_k and $R_{k_{\max}}$, respectively. Both regions are divided into four quadrants. The fraction of fragments for each quadrant is proportional to the SN density of the quadrant itself. If a quadrant does not contain SNs, fragments are not uploaded there; if a quadrant contains more than one SN, the SNs with the largest amounts of available space are chosen as targets for the publication of fragments. Finally, when all target SNs have been chosen, the peer uploads the fragments in parallel.

This process can be generalized. Instead of looking in its neighborhood, the publisher may search for SNs around any remote location in the known map. Therefore, it is possible to implement a publication strategy that takes into account the content that is published. If the content refers to location $\mathcal{L}$, it will be stored nearby $\mathcal{L}$, regardless of the publisher's location. Remote publication may be time-consuming, depending on the node spatial density between the publisher and the target location. Low spatial density may, in fact, lead to long and twisted query propagation processes. The search process, described in the succeeding text, is constrained to a limited region between the location of the searcher and the location of interest $\mathcal{L}$.

3.4. Search process

To search for a data item, a node sends query messages (whose structure has been described in Subsection 3.1) in the direction of the region of interest, to all nodes within an angular sector with variable angle α (dimension: [degrees]). In response to such query messages, asked nodes return

DOI: 10.1002/cpe

Table III. Publication and search parameters.

R_k	Radius of the first geobucket (dimension: [km])
$R_{k_{\max}}$	Radius of the most external geobucket (dimension: [km])
$\mathcal{L}$	Location of interest (geographic coordinates)
α	Angle of the angular sector where a search takes place (dimension: [degrees])

Table IV. RNC parameters.

$\mathcal{M}$	Size of a generic file (dimension: [Bytes])
N_g	Number of generations (dimension: [gen])
N_f	Number of fragments in one generation (dimension: [frag/gen])
s_i	i-th fragment
d_f	Fragment size (dimension: [Bytes/frag])
GF(q)	Galois field of size q (adimensional)
K	Overhead factor (adimensional)
R_c	Coding rate (adimensional)
τ	Guard threshold (dimension: [frag])
ϵ	Fraction of the number of coded fragments (adimensional)
n	Number of coded fragments (dimension: [frag])
T_M	Maintenance period (dimension: [s])

RNC, randomized network coding.

lists of SNs that are aware of the searched data item (because they own some fragments). The angle α of the search angular sector can be increased if the dimension of the retrieved list of nodes is not satisfactory. In this paper, we assume that α can vary between $\alpha_{\min} = 30°$ and $\alpha_{\max} = 60°$. Although these values may be optimized through proper analysis or simulations, they have been heuristically chosen so as to produce at least sufficient results in a smart city scenario. In fact, our simulations suggest that a node looking in such a region is able, through the proposed approach, to recover the information of interest in its application.

After a short time, the node that started the request propagation has a list of N_k SNs that are aware of the data item of interest. It is then sufficient to contact a subset of such SNs to rebuild the original data item, as discussed in detail in Subsection 3.5. If, otherwise, the publisher is not online, the searcher can collect a number of fragments from the SNs that have declared to be aware of the data item. Our system is robust against churns, because new fragments of the same data item are periodically generated (until the time validity of the data item expires) according to the strategies described in Subsection 3.5.

The process is illustrated in Figure 3 (right). Like the publication process, also the search process can be generalized, that is, instead of looking in its neighborhood, the searcher may look for peers around any remote location in the known map. The publication and search parameters are summarized in Table III.

3.5. Randomized network coding strategy

In the following, we detail the RNC operations performed during the publication, retrieval, and maintenance of a given resource in the network. For the sake of clarity, the parameters of the RNC strategy are listed in Table IV.

A file of size $\mathcal{M}$ (dimension: [Bytes]), which needs to be stored, is divided into N_g (dimension: [gen]) generations composed of N_f fragments (dimension: [frag/gen]) $\{s_i\}_{i=1}^{N_f}$ each, so that $\mathcal{M} = N_g N_f d_f$, d_f being the size of each fragment (dimension: [Bytes/frag]).[‡] We now focus on a single generation, because all the operations are the same for each generation. The fragments of a generation can be interpreted as symbols in the Galois field GF(q) and are linearly combined in

[‡]Because all fragments are supposed to have the same size, if a generation is composed by less than N_f fragments, zero padding is applied.

DOI: 10.1002/cpe

order to obtain the coded fragments. The number of linearly encoded fragments is equal to $n = KN_f$ (dimension: [frag]), K being the overhead factor (adimensional). This corresponds, from a coding theory perspective, to a coding rate $R_c = 1/K$ (adimensional). In the presence of RNC, each coefficient of the linear combinations is uniformly chosen among all possible values in GF(q). This implies that there exists a nonzero probability that two coded packets are linearly dependent. However, it is well known that this probability is basically zero if q is sufficiently large [22]. Each packet flowing in the network contains a coded payload and a header, which carries information about the generation (which the fragment belongs to) and the global coding vector of dimension N_f (i.e., the coefficients representing the linear combination of the original symbols). This has been shown to introduce small overhead for practical packet sizes [23].

To retrieve a published file, it is necessary to obtain, for each generation, using the previously illustrated lookup functionality, N_f linearly independent fragments. After these fragments have been collected, a system of linear equations can be solved using the classical Gauss elimination algorithm.

The use of RNC can be taken into account in order to perform the following novel *proactive* resource maintenance strategy. Whenever a UN retrieves a resource, it generates a given number of new fragments for each generation, also checking for network dynamic condition evolution. In particular, the more dynamic the network conditions (e.g., large churns), the larger the number of generated fragments, in order to make the scheme more robust against nodes' failures. In addition to this strategy, resource maintenance can also be carried out *reactively*, by periodically (i.e., every T_M seconds) checking the availability, in the SNs, of the resources. If, for each generation, the percentage of surviving fragments falls below a properly defined retrieval 'guard threshold' (which represents the fraction of fragments below which the resource is likely to become soon unavailable), the node responsible for that resource generates new fragments independent of the surviving ones, and distributes them in SNs. The threshold is denoted as τ (dimension: [frag]) and is equal to a fraction of the total number of fragments, that is, $\tau \triangleq \epsilon n$, $\epsilon \in [1/K, 1]$ (adimensional). The number of new generated fragments is chosen so that the overall number of available fragments for a generation is equal to n, that is, the published number of fragments.

Note that, in the proposed system, proactive maintenance is performed only when a client has finished a successful download, and, therefore, a resource may not be regularly maintained if it is not sufficiently 'popular'. However, the complexity of the maintenance operations can be reduced significantly with respect to the reactive approach. Moreover, as the number of exchanged control messages is significantly smaller, the bandwidth waste is lower. Therefore, an enhanced solution may combine these two techniques in order to obtain an efficient maintenance also for non popular resources. In this case, in fact, resource is reactively maintained by generating new fragments when the resource is downloaded. However, centralized maintenance is also performed with period T'_M (dimension: [s]) so that a sufficient amount of redundancy is guaranteed for all resources. Note that, in general, $T'_M \neq T_M$.

In the following, we will refer to the reactive strategy as 'periodic maintenance' (PM), whereas the new proactive strategy will be denoted as 'sporadic maintenance' (SM). The intermediate strategy will be finally denoted as 'hybrid maintenance' (HM). Note that, in all cases, according to the taxonomy in [7], the maintenance strategy aims at performing *functional repair* and not *exact repair*. For all considered strategies, in fact, the new generated fragments are not exactly the same as those lost by disconnected nodes, but they exhibit the same statistical characteristics.

4. ANALYTICAL SYSTEM PERFORMANCE EVALUATION FRAMEWORK

In this section, we describe an analytical framework to characterize system performance, anticipating some of the results of simulations and experiments later shown in Sections 5 and 6, respectively. Such a model relies on control parameters, which are independent from the network size. Thus, it will be useful to predict the system behavior where lengthy simulations or large-scale experiments would be needed or even impossible to perform. In particular, we characterize the performance of the overlay architecture in terms of hops needed to publish and/or recovery information. The analytical performance evaluation of the coding strategy, taking into account resource availability, is not considered, as this was already investigated in our previous works [17, 24].

DOI: 10.1002/cpe

Therefore, let us now compute the probability mass function (PMF) of the number of hops H needed to find a given number, denoted as N_s, of storage nodes — $H \geqslant 0$, and we refer to the case with $H = 0$ as the initial number of known storage nodes (to account for a possible initial knowledge). To this end, we first evaluate the CDF of H, that is,[§] $F_H(h) = P\{H <= h\}$. Note that $F_H(h)$ can be equivalently reinterpreted as the probability of finding N_s storage nodes with no more than h hops. The following assumptions are expedient for our analysis:

- The request issuer initially knows $0 \leqslant k_s(0) < N_s$ storage nodes.
- The request is sent to the m neighbors in the first geobucket.
- Each request copy is then possibly propagated to one other node, among those that have not yet received the request.
- Each node other than the request issuer returns, on average, n_s different storage nodes not known to the request issuer.

The number of storage nodes known after propagating the request on h hops is denoted as $k_s(h)$ ($k_s(h) \leqslant N_s$), where we have explicitly indicated the dependence on h, and can be computed according the following recursive equation:

$$k_s(h) = k_s(h-1) + \Delta_h$$

where

$$\Delta_h = \left[1 - \frac{k_s(h-1)}{N_s}\right] m \cdot n_s \tag{2}$$

is the increment of knowledge after propagating requests over the h-th hop. The rationale behind this choice is due to the fact that the function should be decreasing with the number of hops and, when $k_s(h) = N_s$, no further increment of knowledge is possible.

Equation (4) can be equivalently rewritten as

$$\begin{aligned} k_s(h) &= k_s(h-1)\left[1 - \frac{m \cdot n_s}{N_s}\right] + m \cdot n_s \\ &= B k_s(h-1) + C \end{aligned}$$

where

$$\begin{aligned} C &= m \cdot n_s \\ B &= 1 - \frac{C}{N_s}. \end{aligned}$$

Expanding the recursive equation leads to

$$\begin{aligned} k_s(h) &= B^h k_s(0) + (B^{h-1} + ... + B + 1)C \\ &= B^h k_s(0) + C \sum_{i=0}^{h-1} B^i. \end{aligned}$$

Finally, observing that $B < 1$, the CDF of H can be written as

$$F_H(h) \triangleq \frac{k_s(h)}{N_s - k_s(0)}. \tag{3}$$

[§]We remark that the CDF $F_H(h)$ is, correctly, a conditional CDF, as it is conditioned on the number N_s of storage nodes. For the sake of notational simplicity, we simply use the notation $F_H(h)$, rather than $F_H(h|N_s)$. The same comment applies also to the corresponding PMF.

DOI: 10.1002/cpe

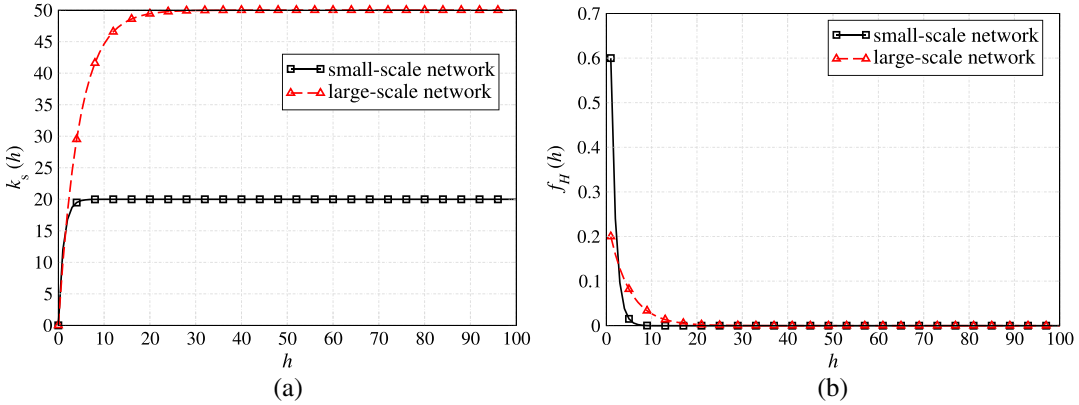

Figure 4. Analytical performance, as a function of the number of hops during publish, for the two considered settings in terms of (a) $k_s(h)$ and (b) $f_H(h)$.

At this point, the PMF of the number of hops needed to find N_s storage nodes can be written, using an incremental approach, as

$$\begin{aligned} f_H(h) &= F_H(h) - F_H(h-1) \\ &= \frac{k_s(h) - k_s(h-1)}{N_s - k_s(0)} \\ &= \frac{[B^h - B^{h-1}]k_s(0) + CB^{h-1}}{N_s - k_s(0)}. \end{aligned}$$

Note that we divide by $N_s - k_s(0)$ so that the probability we are looking for does not depend on the initial conditions $k_s(0)$. Otherwise, the PMF would not be properly defined and the sum would not be equal to 1.

We now consider two scenarios to verify the effectiveness of our approach: (i) a small-scale scenario with $m = 3$, $n_s = 4$, and $N_s = 20$ and (ii) a large-scale scenario with $m = 10$, $n_s = 1$, and $N_s = 50$. In all cases, $k_s(0) = 0$, that is, the publisher has no initial knowledge of any storage node. In Figure 4, we show the analytical performance, as a function of the number of hops during the publication process, for both considered settings in terms of (a) $k_s(h)$ and (b) $f_H(h)$. One should observe that the number of discovered storage nodes increases with the number of hops and obviously saturates to its maximum value. Moreover, the probability of finding new storage nodes decreases and it is verified that

$$\sum_{i=1}^{\infty} f_H(i) = 1$$

that is, $f_H(h)$ is a correct PMF. From these results, one can compute the average number of hops to discover N_s storage nodes, which are equal to

$$\begin{aligned} \mathbb{E}[H] &= \sum_{i=1}^{\infty} i f_H(i) \\ &\simeq \begin{cases} 1.67 & \text{for small-scale networks} \\ 5 & \text{for large-scale networks.} \end{cases} \end{aligned}$$

This shows that the average number of hops can be maintained to sufficiently small values, in the publication process.

DOI: 10.1002/cpe

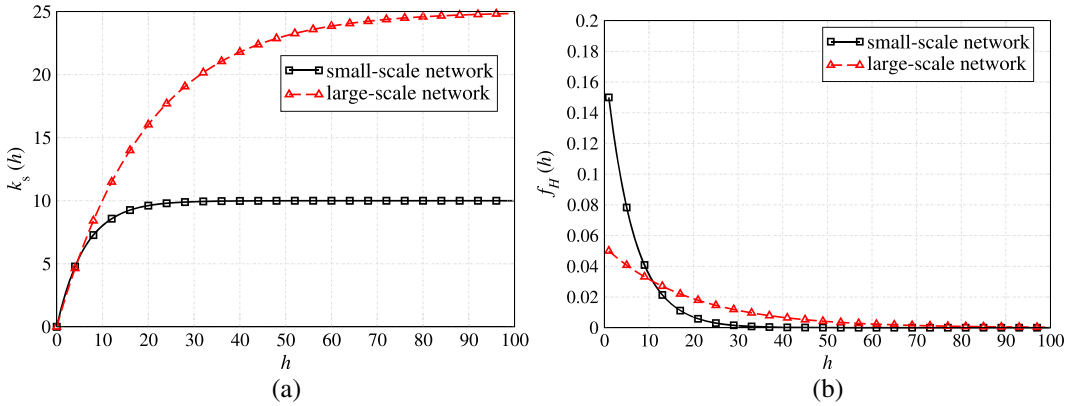

Figure 5. Analytical performance, as a function of the number of hops during search, for the two considered settings in terms of (a) $k_s(h)$ and (b) $f_H(h)$. The value of α is set to 45°.

Finally, note that the same approach can be applied to the other process, that is, data item search. In this case, nodes that know where the desired resource is located are searched within a region whose area is $\alpha/2\pi$ of the whole region of interest. Therefore, one can apply the same model used for the publication process, with the following differences:

- each node targets $m' = (\alpha/2\pi)m < m$ neighbors to send the lookup request;
- each node looks for $N_k < N_s$ storage nodes, which are aware of the resource to be found—usually, $N_k = N_s/K$, where K is the overhead factor introduced in Section 3.5.

In Figure 5, we show the analytical performance, as a function of the number of hops during search, for both considered settings in terms of (a) $k_s(h)$ and (b) $f_H(h)$. The value of α is set to 45°. The same considerations given in Figure 4 for the case of resource publish can be carried out in this case as well. Moreover, assuming $K = 2$, the average number of hops to discover N_k storage nodes is equal to

$$\mathbb{E}[H] = \sum_{i=1}^{\infty} i f_H(i) \simeq \begin{cases} 6.67 & \text{for small-scale networks} \\ 19.92 & \text{for large-scale networks .} \end{cases}$$

The search process requires on average, a higher number of hops with respect to the publish operation, due to the fact that search is focused in a fraction of the entire area. However, the average number of hops is still sufficiently small. Thus, we can conclude that the proposed approach scales well with increasing network sizes, in general.

5. SYSTEM PERFORMANCE EVALUATION

In this section, we analyze the performance of the proposed architecture through discrete event simulations carried out with DEUS, an open source tool that provides a simple Java API for the implementation of nodes, events and processes, and a straightforward (yet powerful) visual editor for configuring simulations [25].

A sensor network deployed in the city of Parma has been simulated, considering n_u UNs that move over realistic paths of length equal to $l = 100$ km, generated using Google Maps API. In this case, the user density is $\delta = n_u/l$. Each simulated UN selects a different path and starts moving over it. Moreover, we have placed (with uniformly random spatial distribution) RDSs and SNs over all the map. ADSs have not been considered, because they would not be different from RDSs (in publishing data) and UNs (in retrieving data). With the features provided by Google API, we have

DOI: 10.1002/cpe

Table V. Main simulation parameters.

Number of SNs	50
Number of RDSs	500
$R_{k_{\max}}$	2.5 km
Search angle α	30° (first attempt) and 60° (other attempts)
Disk space for each SN	Uniformly between 10 and 100 GB
Resource size $\mathcal{M}$	Uniform in [1, 35] MB
Simulated time interval	10 h
Resource search period T_s	3 min
Path length l	100 km
Number of independent trials	20

RDS, raw data source; SN, storage node.

created a simple HTML&Javascript control page, which allows to monitor the temporal progression of the simulated system, with the possibility to select any node and visualize its neighborhood (videos are available at [27]).

The considered setup is summarized in Table V. The network is composed of 50 SNs and 500 RDSs. The number of randomly placed UNs is such that a given linear density of δ (nodes/km) over the roads is obtained. In particular, a path length of 100 km is considered, and δ varies from 5 to 40 so that

$$\#\text{ of UNs} = \delta \times l = 100\delta \in [500, 4000].$$

The storage space at each SN is a uniformly distributed random variable between 10 and 100 GB. As in [20], each node covers a region of interest of size $A = 20\ \text{km}^2$ so that the maximum radius is

$$R_{k_{\max}} = \sqrt{\frac{A}{\pi}} \simeq 2.5\ \text{km}$$

and a dynamic discovery period ranging from 1.5 to 6 min, depending on the number of discovered nodes. The more the geobuckets are filled, the more the discovery period is high. The system parameters for the coding and maintenance strategies are set as in [17]. The lifetime of the system is set to 10 h. Within the first 4 h, the overlay is initialized and stabilized, whereas from the fifth hour RDSs begin publishing data resources. From the sixth hour, UNs start searching contents at random locations implemented using a Poisson process with a mean arrival of approximately 40 s. Moreover, in this last phase storage, nodes are randomly disconnected, according to the following different scenarios:

1. Burst of SN disconnections during the sixth hour of the observed period: at the end of the sixth hour, the number of SNs has been reduced to 10.
2. SN disconnections over the second half of the observed period: the final number of SNs is 10 as in the first scenario.
3. Continuous disconnections and reconnections of SN modeled with a Poisson process with a mean arrival equal to 1.5 min.

Simulation results have been averaged over 20 independent simulation runs in order to reduce statistical fluctuations.

Regarding data publication and search, the following performance metrics are of interest:

- average number of hops per publication;
- average number of exchanged messages per publication required to find an SN;
- average number of hops per search;
- average number of exchanged messages per search.

DOI: 10.1002/cpe

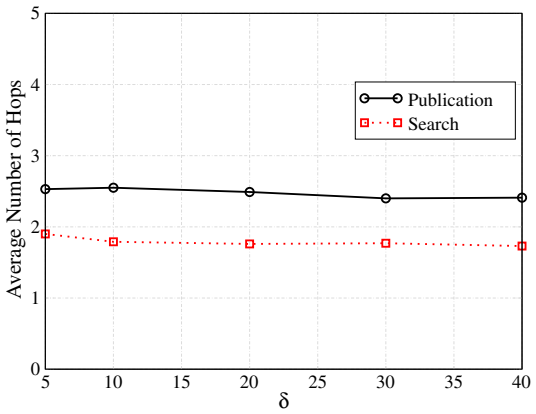

Figure 6. Average number of message propagations (hops) to discover storage nodes (SNs) for publishing or searching data as functions of the user node (UN) density.

The impact of coding, instead, has been evaluated through the following performance metrics:

- resource availability, defined as the probability that a given resource in the network can be reconstructed;
- average used storage space on each SN (dimension: [GB]).

Unfortunately, each distributed geo-referencing information management system has its own parameters and performance indicators, and, therefore, it is difficult to make a fair comparison with our system. As an example, the Globase.KOM architecture proposed by Kovacevic *et al.* [15] apparently needs more hops and shows higher delay than our approach. However, their simulated scenario is different (and impossible to be reproduced) with respect to ours and, therefore, the comparison may not be meaningful.

5.1. Publication and search

As described in Section 3, the publication and search processes are distributed among all types of peers that participate in the DGT. For all simulated scenarios, all lookups were successful, thus confirming the effectiveness of the DGT. In Figure 6, the average numbers of message propagations (hops) to discover SNs, for publishing or searching data, are shown as functions of δ. The value of the overhead factor is $K = 2$. It can be observed that these average numbers of hops are almost constant with respect to the density of UNs. We remark that RDSs are initially aware of a few SNs; however, when their knowledge increases over a threshold (that we set to 5), they stop performing storage node lookups. The average number of publication hops is always smaller than the average number of lookup hops. This is due to the fact that, in order to discover SNs for publishing data, each RDS considers the whole circular region surrounding itself. Instead, the search process covers a restricted region defined by the angle α. These results are in agreement with our analytical framework shown in Section 4. In fact, in the considered settings $k_s(0) = 0$. As we set $N_s = 5$, thus $N_k = 5/2$. In fact, the overall considered area is $10 \times 10 = 100$ km^2, whereas each node sees a circular area with radius $R_{k_{max}} = 2.5$ km, which corresponds to an area approximately equal to 20 km^2, that is, 1/5 of the overall area. Therefore, each node may know, on average, 10 SNs. On the average, in the first geobucket, there are $m = 5$ neighbors,[¶] and, because each of them has the same coverage area, they can know on average 5 SNs. Therefore, it follows that $n_s = 1$, but we set $n_s = 0.5$ to be more conservative (for instance, mobility may decrease this value). Using the model, it follows that

[¶]We measured the value $m = 5$ by means of simulations, considering the urban scenario described in this section. Such a value is almost independent from δ.

DOI: 10.1002/cpe

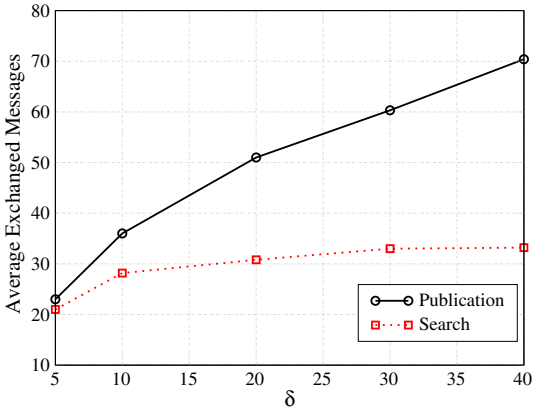

Figure 7. Average number of exchanged messages for publication and search as functions of the user node (UN) density.

$$\mathbb{E}[H] \simeq \begin{cases} 2 & \text{for publication} \\ 6 & \text{for conservative search.} \end{cases}$$

Note that the number of hops for search is highly overestimated, due to the fact that the conservative values $k_s(0) = 0$, $n_s = 0.5$, and $\alpha = 60°$ are considered. If $n_s = 1$ and $\alpha = 30°$, $\mathbb{E}[H] = 3$, which is closer to the real value estimated by simulations. Thus, for the search process, we can consider the analytical result as an upper bound.

Also, note that the simulation results do not depend on the linear density δ of UNs. Indeed, the size of the region where each node performs publication and search is sufficiently large to allow good performance even for small values of δ. Larger values of linear density do not affect m so much, for which results do not change.

In Figure 7, the average numbers of exchanged messages, for publication and search, are shown as functions of δ. As in Figure 6, the performance for search is and always smaller than for publish. However, the curves increases with δ, because the more messages are exchanged in the presence of a larger linear spatial densities and, therefore, numbers of UNs.

5.2. *Storage node disconnections*

Figure 8 shows the resource availability, as a function of time, for the first scenario (burst of SN disconnections), for $K = 2$ and $K = 5$. The PM and HM maintenance strategies are considered with different values of T_M. Note that only one curve is associated with SM, because our results show that the frequency of the search process has a minor impact on the performance. One can observe that, in correspondence of the disconnection burst, the resource availability reduces. At this point, the availability starts growing again, because the maintenance process allows the introduction of new fragments in substitution of the lost ones. If $K = 2$ is considered, all strategies exhibit similar performance. On the other hand, if the publication overhead is higher ($K = 5$), a different performance can be observed. In this case, in fact, each resource is encoded with a larger number of fragments, and, therefore, the system is more robust against nodes' failures. Note that PM has better performance than SM for either $T_M = 30$ or $T_M = 120$ minutes. In fact, with SM only popular resources are maintained and, therefore, the average resource availability is not satisfactorily in the presence of SN disconnections. Other results, not shown here for the ease of clearness, show that SM starts to be preferable for larger values of the maintenance period, for example, $T_M = 180$ min. This is due to the fact that with larger maintenance periods there is a higher probability that disconnections happen, thus causing losses of fragments and, therefore, of entire resources. In all cases, HM outperforms both strategies, because it combines SM and PM. Note that the scope of this analysis is to show the potential benefit brought by the combined use of PM and SM. However, one may optimize system parameters, that is, to determine the value of T' such that HM has exactly the same performance of PM. However, this goes beyond the scope of this paper.

DOI: 10.1002/cpe

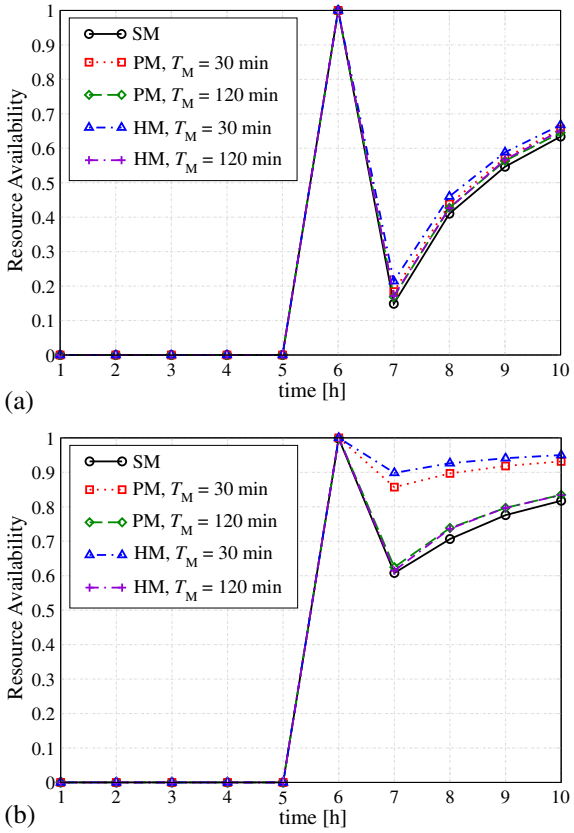

Figure 8. Resource availability, as a function of time, in the first scenario with a burst of storage node (SN) disconnections for (a) $K = 2$ and (b) $K = 5$. The three different maintenance strategies (periodic maintenance (PM), sporadic maintenance (SM), and hybrid maintenance (HM)) are considered with different values of T_M.

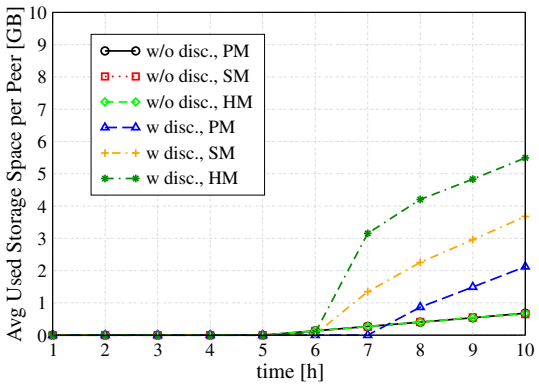

Figure 9. Average occupied storage space per node, as a function of time, with sporadic maintenance (SM), hybrid maintenance (HM), and periodic maintenance (PM) (with $T_M = 30$ min) and $K = 5$. The performance in the presence of the burst of SN disconnections is compared to the case of no disconnections.

In Figure 9, the average occupied storage space per node is shown—as a function of time—with SM, HM, and PM (with $T_M = 30$ min) and $K = 5$ (from Figure 8, with $K = 2$ no performance variation can be observed). One can observe that without disconnections, the maintenance strategy has no impact, because it can be shown that the same resource availability can be achieved. In this case, in fact, no maintenance events are scheduled, and, therefore, the increase on the occupied storage

DOI: 10.1002/cpe

space is given by the creation of new resources. When the disconnections happen, instead, the occupied storage space increases. In this scenario, in fact, regeneration is performed, and, therefore, more fragments are inserted in the network. However, the number of SNs decreases, and, therefore, the (less or more) same number of fragments should be placed in a lower number of nodes, thus resulting in an increase in the average occupied disk space. Directly comparing Figure 9 with Figure 8, one can observe that HM achieves a better availability, however, at the price of a larger storage usage. This clearly shows the trade-off between the achievable resource availability and the cost in terms of storage usage. Finally, other results, not shown here for the sake of clarity, show that the storage occupancies are more significant if less space is available (e.g., at most 50 GBs instead of 100). This is not unexpected, as in this case the network is more saturated by encoded fragments.

In Figure 10, the resource availability is shown, as a function of time, in the second scenario with SN disconnections over a long period of time. SM, HM, and PM (with $T_M = 1$ h) are considered. In this case as well, the maintenance processes with HM or PM preserve the resource availability to the maximum value (i.e., 1). In the presence of SM, however, the resource availability strongly decreases during the last hour. The reason is that, the more the number of SN decreases, the higher the probability of file retrieval failure, which makes its reconstruction not possible. Note finally that HM and PM have the same performance due to the fact that SM has no impact on the overall resource availability.

Figure 11 shows the resource availability—as a function of time—in the third scenario with continuous SN disconnections and reconnections, comparing SM, HM, and PM (with $T_M = 1$ h). One should observe that, with PM and HM, the resource availability is not affected by the churn process.

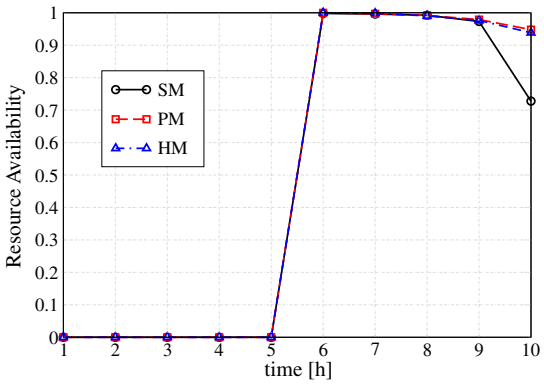

Figure 10. Resource availability, as a function of time, in the second scenario with storage node (SN) disconnections over a long period of time. Sporadic maintenance (SM), hybrid maintenance (HM), and periodic maintenance (PM) (with $T_M = 1$ h) are considered.

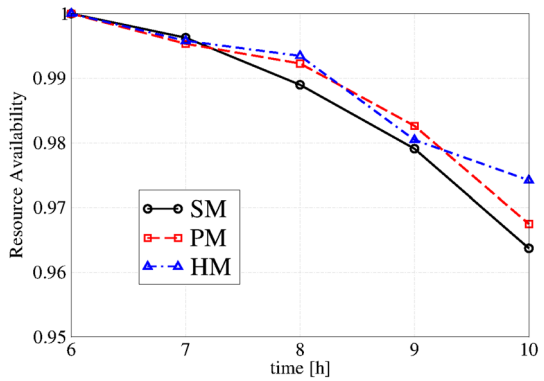

Figure 11. Resource availability, as a function of time, in the third scenario with continuous storage node (SN) disconnections and reconnections, comparing sporadic maintenance (SM), hybrid maintenance (HM), and periodic maintenance (PM) (with $T_M = 1$ h).

DOI: 10.1002/cpe

SM, instead, is not sufficient to preserve resource availability to its maximum value. The reason is that if resources cannot be retrieved, because of SN disconnections, new fragments are not inserted in the system. Moreover, SM is improved by the use of extra periodically scheduled maintenance events. Therefore, the hybrid strategy has to be preferred in scenarios with continuous connections and disconnections of SNs.

6. SIP2PEER-BASED IMPLEMENTATION

6.1. Implementation

The fact that the DGT routing strategy allows an efficient and robust resource publish/retrieval, has been shown, in previous sections, by means of a simple (yet insightful) analytical model and simulations. However, a realistic on-field experimentation would be required to further validate the idea, although it is difficult because too many users are needed. In this section, we present a Sip2Peer-based prototype, which implements the DGT approach. The coding strategy, that is, the use of RNC is actually not implemented and will be matter of future work.

The DGT Java Library implements the basic functionalities of the DGT overlay architecture, that is, discovery, neighborhood management, and geobucket maintenance. This implementation is based on the Sip2Peer open source middleware [26], which is gaining momentum within the P2P community. Sip2Peer is based on the Session Initiation Protocol [28] for the implementation of any P2P or generally distributed application or overlay, without constrains on the peers' nature (they may run not only on traditional PCs but also on mobile nodes), and specific architecture.

A logical description of the Sip2Peer-based DGT implementation is shown in Figure 12. At the lower layer, there is the Sip2Peer layer, where the basic Session Initiation Protocol functionalities are given. On top of this layer, we have implemented all the functionalities for handling the DGT messages. Based on this level, the main DGT aspects have been implemented. In particular, the upper level is composed of the DGT kernel, the geobucket manager, and the location service manager. Moreover, on top of the DGT-based functionalities, the publication/retrieval processes have been implemented. In the current implementation, neither RNC strategies have been considered nor maintenance aspects, being this is the matter of future work. Therefore, the proposed experimental activity is geared toward a preliminary assessment of the effectiveness of the proposed DGT-based architecture in a realistic environment.

The basic DGT functionalities reflect the approach proposed in Section 3. Behind the publication and retrieval operations of the city management application, we have implemented information dissemination operations, in the context of smart city scenarios. This can be also useful in other sensing applications where the same information should be given to the same area from different RDSs, which, however, observe the same phenomenon.

A few implementation issues have been tackled to optimize the basic implemented library—in particular, to reduce the amount of messages exchanged among the network. The first optimization is related to the number of exchanged messages during the publication and discovery phases. Each

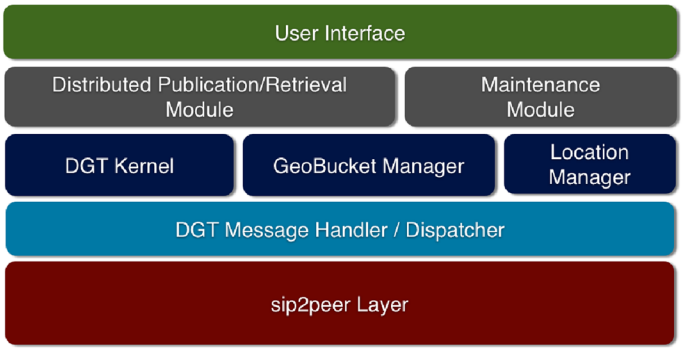

Figure 12. Modules of the DGT Sip2Peer-based implementation.

DOI: 10.1002/cpe

DGTEvent message is attached with a list of size L containing the descriptors of peers which have already received that message. In this way, the message is forwarded only to nodes interested to it *and* not already receiving it. Moreover, the messages are forwarded by a peer to its neighborhood with a given probability p. It is straightforward to observe that the forwarding probability has to be properly set. In fact, if this parameter is too low, a given message may be stopped somewhere in the network, and no more propagated. Another implemented modification is a method to inform when a node is disconnected from the network. In the original solution, a node is recognized as disconnected if it does not reply to a presence message. In the modified solution, instead, if a node wants to disconnect from the network, it informs its neighborhood. Finally, the DGT messages have been provided with a time to live, as well as the possibility to reduce the list size from the bootstrap node. Also, these optimizations have been made to reduce the bandwidth usage. A final important feature, inherited from Sip2Peer, is the Session Border Controller (SBC)-based network address translation (NAT) traversal mechanism, which allows the peers to communicate in presence of NATs—a very relevant problem especially in mobile scenarios. The SBC is a node provided with a public IP address. It allows a generic peer to check if it is in a private network with a NAT, also requesting (if needed) an IP address/port pair to communicate with other peers.

6.2. *Experimental evaluation*

The developed library has been initially tested both on traditional PCs and on real Android devices to validate the designed behaviors with different mobility patterns and network coverages. Consecutively, a preliminary evaluation of the DGT Java Library has been carried out by deploying 200 peers on a cluster at our Department. As in the simulation-based performance evaluation reported in Section 5, each peer is randomly placed in the area of the city of Parma. All peers are connected to the DGT overlay network and can move according to the FTM mobility model described by (1). Performance evaluation is performed by letting each peer logging every 30 s its speed, geographical position, and amount and type of exchanged data. During network operations, another peer is generated and selected as the bootstrap node, which generates random events, for example, accidents or road works. Three scenarios have been analyzed, changing L and p. In the first case, $L = 0$ and $p = 1$; in the second scenario, $L = 15$ and $p = 1$; in the third scenario, $L = 0$ and $p = 0.5$. The following performance metrics have been evaluated:

- *Bandwidth*, denoted as B, expressed in terms of average kB/s of data sent by a peer.
- *Delay*, denoted as D, expressed as the average time required to deliver an event message and to receive the acknowledgement reception.
- *Number of neighbors*, denoted as NN, expressed as the average current number of neighbor of a peer.

Finally, five independent runs have been performed to average over statistical fluctuations.

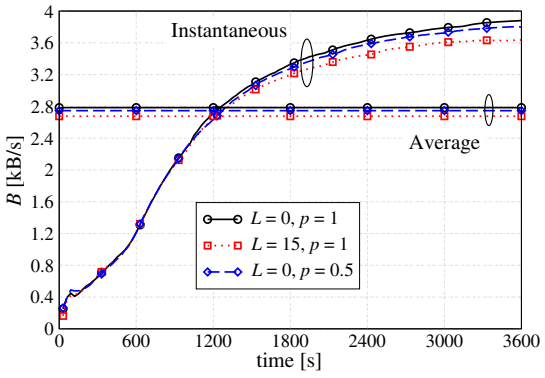

Figure 13. Bandwidth B (dimension: [kB/s]), as a function of time, for the three considered scenarios. Both time-instantaneous and average values are shown.

DOI: 10.1002/cpe

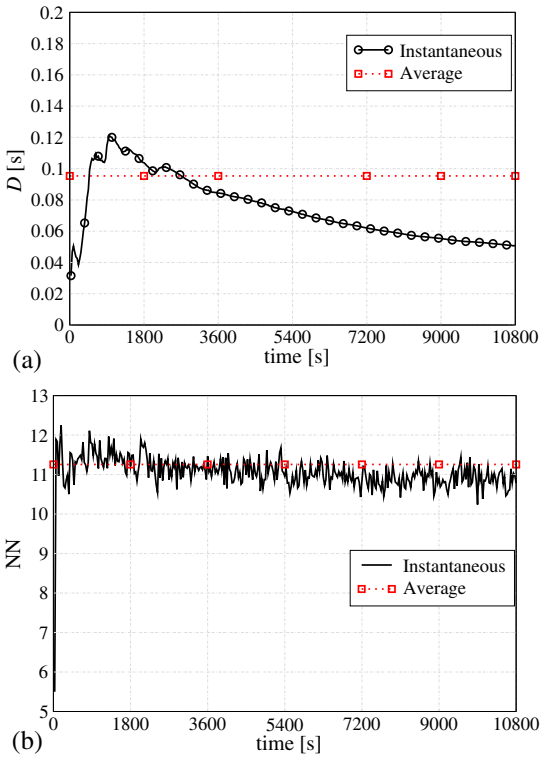

Figure 14. (a) Delay (D) and (b) average number of neighbors (NN), as functions of time, for $L = 0$ and $p = 0.5$. Both time-instantaneous and average values are shown.

In Figure 13, B (dimension: [kB/s]) is shown, as a function of time, for the three considered scenarios. Both time-instantaneous and average values are shown. One can observe that the bandwidth increases with time, because more and more packets and messages are generated during the experiments. For a fixed value of p, increasing the value of L leads to better performance. In fact, each forwarding node is aware of more nodes which have already received a given message. Moreover, for a fixed value of L, decreasing the forwarding probability p reduces the bandwidth usage, due to the fact that less packets are transmitted across the network.

Because from Figure 13 the lowest bandwidth usage can be obtained with $L = 0$ and $p = 0.5$, we now analyze the delay and neighboring performance in this scenario. In Figure 14, (a) D and (b) NN are shown, as functions of time, for $L = 0$ and $p = 0.5$. Both time-instantaneous and average values are shown. Note that both D and NN are stable around their mean value, thus showing that the effectiveness of the proposed DGT approach is confirmed in the experimental setting as well.

7. CONCLUDING REMARKS

In this paper, we have proposed a joint DGT/RNC architecture for the robust management of large amount of information in smart cities. We have illustrated the DGT overlay scheme and a routing strategy for building peer neighborhood, as well as publishing and retrieving data items. Moreover, we have envisioned different resource maintenance strategies to overcome data losses. The use of RNC as a viable and effective technology for distributed storage is not novel, but the novelty lies in the geo-localized mixing of such pillars—P2P message routing and network coding.

Performance has been characterized by means of analysis, simulations, and realistic experiments. The analytical framework allows to characterize the performance of the DGT overlay scheme, showing that a few hops are needed to publish and search processes. Simulations of dynamic scenarios in a realistic environment (the city of Parma) have shown that the system is robust against SN

DOI: 10.1002/cpe

disconnections. Even in extreme conditions (e.g., 80% of SNs disconnected), maintenance strategies guarantee high resource availability. In particular, we have shown that PM performs better than SM if maintenance is sufficiently frequent. On the other hand, SM is effective only with popular resources, because regeneration is performed only after a successful download. To overcome this problem, we have proposed to combine these two strategies. Results show a resource availability improvement, at the expense of higher storage occupancy.

We have also proposed a partial implementation based on the Sip2Peer middleware to the purpose of achieving an experimental validation of the proposed system. This preliminary version only implements the overlay architecture, without any form of redundancy generation. Preliminary tests appear promising for the our goal of proving the effectiveness of the approach on the field.

Future work will be devoted to the design of a maintenance strategy based on network state analysis and to the implementation of the proposed architecture. Moreover, the experimental Sip2Peer-based testbed will be extended to take into account the network coding strategy, as well as resource maintenance. This will allow to perform on-field testing, in order to find an optimal tuning of the (several) control parameters, which characterize the proposed solution. In a real deployment, we think that one major issue will be network handover (both vertical and horizontal). While developing the software, we are taking this aspect into account.

ACKNOWLEDGEMENTS

We would like to thank Giacomo Brambilla, from Università degli Studi di Parma, Italy, for his help in the implementation of the proposed solution using the Sip2Peer middleware.

REFERENCES

1. Steventon A, Wright S (eds.) *Intelligent Spaces: The Application of Pervasive ICT*. Springer-Verlag: London, UK, 2006.
2. California Institute for Smart Communities. *Ten Steps to Becoming a Smart Community*, 2011.
3. Santa J, Moragon A, Gomez-Skarmeta AF. Experimental evaluation of a novel vehicular communication paradigm based on cellular networks. *IEEE Intelligent Vehicles Symposium*, Eindhoven, Netherlands, 2008; 198–203.
4. Jedrzej A, Scheuermann B, Koegel M, Mauve M. PeerTIS: a peer-to-peer traffic information system. *Proceedings of the ACM International Conference on Mobile Computer and Networking* (MOBICOM), Beijing, China, 2009; 23–32.
5. IBM. *SmarterCity Initiative*. (Available from: www.ibm.com/thesmartercity).
6. Picone M, Amoretti M, Zanichelli F. GeoKad: A P2P Distributed Localization Protocol. *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications (PerCom 2010)*, 2010; 800–803.
7. Dimakis AG, Godfrey PB, Wu Y, Wainwright MO, Ramachandran K. Network coding for distributed storage systems. *IEEE Transactions on Information Theory* 2010 September; **56**(9):4539–4551.
8. Dimakis A, Godfrey PB, Wainwright MJ, Ramchandran K. The benefits of network coding for peer-to-peer storage Systems. *Proceedings of thw International Symposium on Network Coding* (NetCod), San Diego, CA, USA, 2007; 1–6.
9. Li B, Niu D. Random network coding in peer-to-peer networks: from theory to practice. *Proceedings of the IEEE* 2011; **99**(3):513–523.
10. Amoretti M. A survey of peer-to-peer overlay schemes: effectiveness, efficiency and security. *BSP Recent Patents on Computer Science* 2009; **2**(3):195–213.
11. Oggier F, Datta A. Byzantine fault tolerance of regenerating codes. *Proceedings of the IEEE International Conference Peer-to-Peer Computing* (P2P), Kyoto, Japan, 2011; 112–121.
12. Rao W, Vitenberg R, Tarkoma S. Towards optimal keyword-based content dissemination in DHT-based P2P networks. *Proceedings of the IEEE International Conference Peer-to-Peer Computing* (P2P), Kyoto, Japan, 2011; 102–111.
13. Mega G, Montresor A, Picco GP. Efficient dissemination in decentralized social networks. *Proceedings of the IEEE International Conference Peer-to-Peer Computing* (P2P), Kyoto, Japan, 2011; 338–347.
14. Montassier G, Cholez T, Doyen G, Khatoun R, Chrisment I, Festor O. Content pollution quantification in large P2P networks: a measurement study on KAD. *Proceedings of the IEEE International Conference Peer-to-Peer Computing* (p2p), Kyoto, Japan, 2011; 30–33.
15. Kovacevic A, Liebau N, Steinmetz R. Globase.KOM - a P2P overlay for fully retrievable location-based search. *Proceedings of the IEEE International Conference Peer-to-Peer Computing* (p2p), Galway, Ireland, 2007; 87–96.
16. Ahlswede R, Cai N, Li SYR, Yeung RW. Network information flow. *IEEE Transactions on Information Theory* 2000; **46**(4):1204–1216.

DOI: 10.1002/cpe

17. Martalò M, Picone M, Bussandri R, Amoretti M. A practical network coding approach for peer-to-peer distributed storage. *Proceedings of the IEEE International Symposium on Network Coding* (netcod), Toronto, Canada, 2010; 103–108.
18. Gkantsidis C, Miller J, Rodriguez P. Comprehensive view of a live network coding P2P system. *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, Rio de Janeiro, Brazil, 2006 October; 177–188.
19. Picone M, Amoretti M, Zanichelli F. Evaluating the robustness of the DGT approach for smartphone-based vehicular networks. *IEEE Workshop On User Mobility and Vehicular Networks*, Bonn, Germany, 2011; 820–826.
20. Picone M, Amoretti M, Zanichelli F. Proactive neighbor localization based on distributed geographic table. *Proceedings of the International Conference Advances in Mobile Computing and Multimedia*, 2010; 305–312.
21. Seskar I, Marie S, Holtzman J, Wasserman J. Rate of location area updates in cellular systems. *Proceedings of the IEEE Vehicular Technology Conference* (vtc), Vol. 2, Denver, CO, USA, 1992; 694–697.
22. Fragouli C, Soljanin E. *Network Coding Fundamentals*. Now Publisher Foundations and Trends in Networking: Hanover, MA, USA, 2007.
23. Fragouli C, Soljanin E. *Network Coding Applications*. Now Publisher Foundations and Trends in Networking: Hanover, MA, USA, 2007.
24. Martalò M, Picone M, Amoretti M, Ferrari G, Raheli R. Randomized network coding in distributed storage systems with layered overlay. *Information Theory and Applications Workshop (ITA), 2011*, 2011; 1–7.
25. Amoretti M, Picone M, Zanichelli F, Ferrari G. Simulating mobile and distributed systems with DEUS and ns-3. *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS)*, Helsinki, Finland, 2013; 107–114.
26. Distributed Systems Group. *Sip2peer project home page*. (Available from: http://code.google.com/p/sip2peer/) [Accessed on 25 January 2014].
27. Distributed Systems Group. *Smart City videos*. (Available from: http://dsg.ce.unipr.it/d4v) [Accessed on 25 January 2014].
28. Rosenberg J, Schulzrinne H, Camarillo G, Johnston A, Peterson J, Sparks R, Handley M, Schooler E. RFC 3261 - SIP: Session Initiation Protocol. *Technical Report*, IETF Network Working Group, 2002.

DOI: 10.1002/cpe