A Joint Peer-to-Peer and Network Coding Approach for Large Scale Information Management

Marco Picone[1], Michele Amoretti[2], Marco Martalò[1], Erind Meco[1], Francesco Zanichelli[1], Gianluigi Ferrari[1]
(1) Dipartimento di Ingegneria dell'Informazione, (2) Centro Interdipartimentale SITEIA.PARMA
Università degli Studi di Parma
Parco Area delle Scienze 181/A, 43124, Italy
Email: michele.amoretti@unipr.it

***Abstract*—The widespread availability of connectivity to the Internet allows to share large amount of information generated by the most heterogeneous, possibly mobile, sources. One scenario where this situation arises is given by smart cities, which are envisioned to generate and consume relevant information about their statuses to enhance the security and lifestyle of their citizens. In this context, a very challenging question is how the information can be maintained and distributed among the city itself. In this paper, we propose a system architecture based on the creation of a distributed geographic overlay network, which allows to achieve the desired goals. Moreover, information is redundantly encoded by means of randomized network coding, in order to dynamically and distributedly preserve the resource availability. By means of simulations, we investigate the behavior of the proposed solution, in terms of efficiency and speed in data publication/search, as well as resource availability and storage occupancy requirements.**

***Keywords*—Peer-to-peer (P2P), network coding, smart cities.**

I. 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 there are emerging opportunities to introduce digital nervous systems, intelligent responsiveness, and optimization at every level of system integration. For instance, automobiles can participate in the mobility Internet for sharing traffic and travel data. In this context, one major aspect being discussed in the research community is centralization versus decentralization for robust and secure data storage and retrieval [4]–[6].

The availability of massive amounts of sensed information provides fascinating opportunities to understand city activity by means of modeling and analytics. 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 a proactive or reactive manner, to city actors themselves. In this paper, we propose an efficient scheme for maintaining and distributing large amount of information. In particular, our approach consists in defining exchanging information architectures, e.g., overlay networks, which allow to store and distribute city status information with minimum overhead and high reliability. Data regeneration is also necessary to improve the robustness of the system against possible losses.

To this end, we envision the integration of innovative management networks based on the key concepts of peer-to-peer (P2P) and network coding (NC) [3]. While the former optimizes the load balancing and avoids the presence of bottlenecks and single points of failure, the use of NC techniques leads to the presence of redundancy, which promises to make any information retrieval extremely reliable and real-time streaming highly efficient. Indeed, NC improves the performance of P2P content sharing systems since it mitigates the block transfer scheduling or piece selection problem, especially when nodes dynamically join/depart from the Internet. Moreover, NC is important also for another functionality of the system, i.e., distributed storage: should a storage node fail, the stored information could be retrieved by properly combining the information contained in other storage nodes. Our approach differs from other approaches in the literature, where forward error correction (FEC) coding is used to reliably store data across the network.

In the field of smart cities, our work is the first, to the best of our knowledge, that proposes a completely decentralized P2P/NC-based solution for data storage and retrieval. The architecture includes different types of peers—data sources, storage nodes, data aggregators, user nodes—that may be fixed or mobile. Such peers are organized in a structured overlay scheme called Distributed Geographic Table (DGT), that takes into account their geographic position, thus enabling a number of city-tailored services. A possible application of interest is a traffic information system that supports vehicular mobility, by suggesting alternative routes to avoid traffic jams or accidents. In this scenario, the knowledge of the information geographic position may be useful in computing the routes to be suggested to the users.

The paper is organized as follows. In Section II, we briefly discuss the reference works in the field of P2P and NC. In Section III, our joint P2P/NC-based approach for information maintenance and distribution is illustrated. In Section IV, the performance of the proposed solution is evaluated, through simulations, considering realistic dynamic scenarios. Finally, Section V concludes the paper.

II. 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 [7]. Recent research is focusing on P2P-based large-scale storage systems [8], distributed hash tables [9], social networks [10], and measurements of real systems [12]. Particularly promising are the P2P approaches based on geographic localization, e.g, Globase.KOM [11], and those based on traffic information, such as [5].

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. With NC, in fact, intermediate nodes are not only allowed to forward incoming packets, but also to encode them. This allows to achieve the multicast capacity [13] and, therefore, leads to potential advantages in terms of bandwidth and computational efficiency, robustness, etc. 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 practical scenarios of interest is distributed storage [3]. In our previous work [14], we have illustrated how NC and peer-to-peer can be enabling technologies for robust distributed storage. NC has also been considered in [15], 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.

III. Architecture

In this section, we illustrate our distributed architecture for the management of information flows in smart cities. We first introduce the P2P overlay scheme, denoted as DGT, that 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 used to assure data survival.

A. 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 [7]. 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

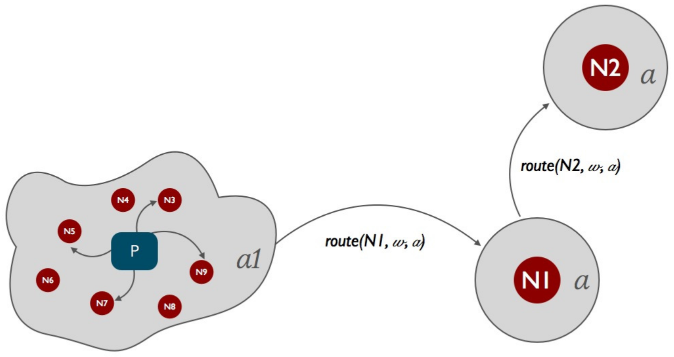

Figure 1: Propagation of a query between nodes to retrieve the neighborhood of a remote region of interest.

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 not excessive.

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 [16], [17]. 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 main DGT concepts are illustrated in details in [16]. In the following we focus on the routing strategy, which is one of the novel contributions of this work.

The DGT routing strategy can be described as a function R that, given a geographic position ω, returns the set of neighbors $N(\omega, a)$ 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 to maintain the neighborhood of a peer, but also to discover active peers in a remote region of interest. In Figure 1, the propagation of a query between nodes to retrieve the neighborhood of a remote region of interest is shown.

The query propagation process is affected by the distance between the source and the destination of the query itself and by 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.

B. Node Description

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

Raw Data Source (RDS) generates basic data pieces and does not usually have local storage capabilities. Usually, it is associated to specific sensors monitoring critical aspects (e.g., traffic cameras, pollution level sensors, etc.). Storage Node (SN) is able to store the (possible large) amounts of data produced by monitoring subsystems. Aggregated Data Source (ADS) is a consumer of data produced by RDS, but also producer of filtered/aggregated data representative of concise "picture" of the city status at a given place and/or time. User Node (UN) is interested in receiving and visualize raw or aggregated data in real time, related to any point of interest.

Each peer has a PeerDescriptor that contains the following information: node type identifier (i.e., RDS, ADS, UN, SN), communication reference (i.e., source/destination IP, port, proxy), prioritized list of locations of interest and data types that the nodes want to proactively receive, and 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 that 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 univocally identifiable by a key that is generated by hashing the item descriptor.

In general, all node types can support mobility. However, in this paper we assume that all nodes are fixed, with the exception of UN that are associated with vehicles moving on streets according to the Fluid Traffic Model (FTM) [18], that fits very well with scenarios characterized by different speed limits for each virtual path. The FTM describes speed as a monotonically decreasing function of vehicular density, forcing lower speed when the traffic congestion reaches a critical point. In particular, the desired speed of a car moving along the ℓ-th possible path is given by

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

where $v_{\text{des}}^{(\ell)}$ is the evaluated desired speed, $v_{\min}$ is the minimum car speed according to vehicle characteristics, $v_{\max}^{(\ell)}$ is the speed limit related to the path, δ_{jam} is the critical vehicular density for which a traffic jam appears, and δ is the current linear density along the road.

C. Publication Process

The process for publishing a data item is carried out as follows. If the publisher does not know enough SNs, or those that it knows are not available, publication request messages, with structure as in Subsection III-A, are propagated in a circular region of radius $R_{k_{\max}}$ (see Figure 2). Each message is propagated avoiding nodes that have already received it. Moreover, the number of propagation hops for each request message is reasonably limited. As a result of this message propagation, the publisher obtains a list of SNs that are interested in maintaining the data item.

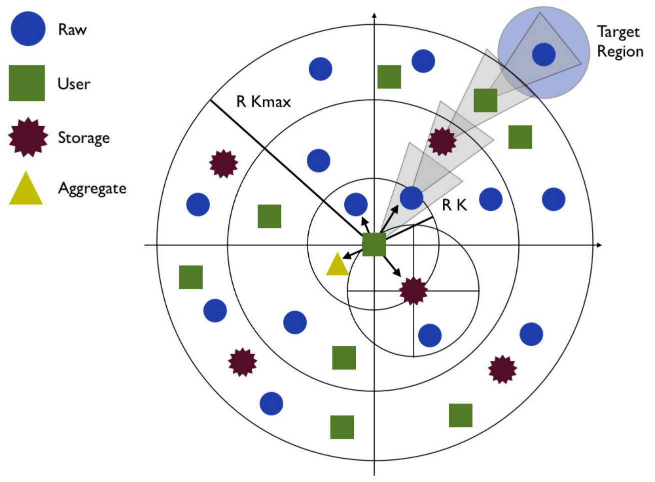

Figure 2: Publication and search processes. In the publication process, messages are propagated in the four quarters of the circular region centered in the node. On the other hand, in the search process, messages are propagated only in the quarter of the target location.

From each data item to be published, a set of fragments is generated, using the randomized NC (RNC) strategy illustrated in Subsection III-E. Such fragments (identified by unique keys and by the key of the data item they have been generated from) are uniformly distributed among peers 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 2. A fraction of the fragments is published in the small circular region with radius R_k. The other fragments are published in the circular crown whose radiuses are R_k and $R_{k_{\max}}$. 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.

Note that, if a quadrant contains more than one SN, those with more 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 a parallel way.

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. Thus, it is possible to implement a publication strategy that takes into account the content that is published. If the content refers to location L, it will be stored nearby L, independently on the location of the publisher. Remote publication may be time-consuming, depending on the density of nodes between the publisher and the target location. Low density may lead to long and twisted query propagation processes. The search process, described below, is constrained to a limited region between the location of the searcher and the location of interest L.

D. Search Process

The process for publishing a data item is carried out as follows. The node generates request messages for a data item described by a DataDescriptor. Such messages (whose structure has been described in Subsection III-A) are sent, in the direction of the region of interest, to all nodes within a conic region with dynamic angle α. Query messages return lists of nodes that are aware of the searched data item. The angle of the search cone can be enlarged if the dimension of this list of nodes is not satisfactory. In this paper, we consider, as reasonable values, $\alpha_{\min} = 30^\circ$ and $\alpha_{\max} = 60^\circ$—future work will be devoted to the optimization of these parameters.

After a short time, the node that started the request propagation has a list of nodes that are aware of the data item of interest. If the node finds the peer that generated the data item in this list, it can obtain the list of SNs that contain the fragments of the data item. It is then sufficient to contact a subset of such SNs to re-build the original data item, as better discussed in Subsection III-E.

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 churn, since 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 III-E.

E. Network Coding Strategy

In the following, we detail the NC operations performed during the publication, retrieval, and maintenance of a given resource in the network.

A file of size $\mathcal{M}$, which needs to be stored, is divided into $N_{\rm g}$ generations composed of h fragments $\{s_i\}_{i=1}^{h}$ each, so that $\mathcal{M} = N_{\rm g} h d_{\rm F}$, $d_{\rm F}$ being the size of each fragment.[a] We now focus on a single generation, since 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 order to obtain the coded fragments. The number of linearly encoded fragments is equal to $n = Kh$, K being the overhead factor. This corresponds, from a coding theory perspective, to a coding rate $R_{\rm c} = 1/K$. 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 non-zero probability that two coded packets are linearly dependent. However, it is well known that this probability is basically zero if q is sufficiently large [19]. Each packet flowing in the network contains both the coded payload and a header, which carries information about the generation, which the fragment belongs to, and the global coding vector of dimension h, i.e., the coefficients representing the linear combination of the original symbols. This has been shown to introduce small overhead for practical packet sizes [20].

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

To retrieve a published file, it is necessary to obtain, for each generation, using the previously illustrated lookup functionality, h 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 churn, 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 be also carried out *reactively*, by periodically (i.e., every $T_{\rm M}$ 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 and is equal to a fraction of the total number of fragments, i.e., $\tau \triangleq \epsilon n$, $\epsilon \in [1/K, 1]$. The number of new generated fragments is chosen so that the overall number of available fragments for a generation is equal to n, i.e., 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, bandwidth waste is lower. An interesting extension, which however goes beyond the scope of this paper, is to combine the two techniques in order to obtain an efficient manteinance also for non popular resources.

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). Note that, in both cases, according to the taxonomy in [3], the maintenance strategy aims at performing *functional repair* and not *exact repair*. In both cases, in fact, the new generated fragments are not exactly the same as those lost by disconnected nodes, but they have the same statistical characteristics.

IV. System Performance Evaluation

In this section, we discuss on the performance of the proposed architecture, that we have carried out by means of discrete event simulations. We have used DEUS, an open source tool that provides a simple Java API for the implementation of nodes, events and processes, and a straightforward but powerful visual editor for configuring simulations [21].

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 $\delta = n_{\mathrm{u}}/l$. Each simulated UN selects a different path and starts moving over it. Moreover, we have placed (with uniformly random spatial distribution) RDS and SN over all the map. ADS have not been considered, since they would not be different from RDS (in publishing data) and UN (in retrieving data).

With the features provided by Google API we have created a simple HTML&Javascript control page that allows us to monitor the temporal progression of the simulated system, with the possibility to select any node and visualize its neighborhood (videos are available at [22]).

The considered set-up is composed by 50 SNs, 500 RDS nodes, and different numbers of UN so that the linear density over the roads δ are equal to 10, 20, 30, and 40 nodes per km. As in [16], each node covers a region of interest of size 19.6 km^2 and a dynamic discovery period ranging from 1.5 min to 6 min depending on the number of discovered nodes. Ten hours of system life are covered. Within the first four hours, the overlay is initialized, whereas during the fifth hour RDS nodes publish 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 seconds. Moreover, in this last period 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 minutes.

Simulation results have been averaged over five independent simulation runs in order to reduce statistical fluctuations. Longer simulations are actually under deployment.

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

- average number of hops per publication;
- average number of exchanged messages for finding SN per publication;
- average number of hops per search;
- average number of exchanged messages per search.

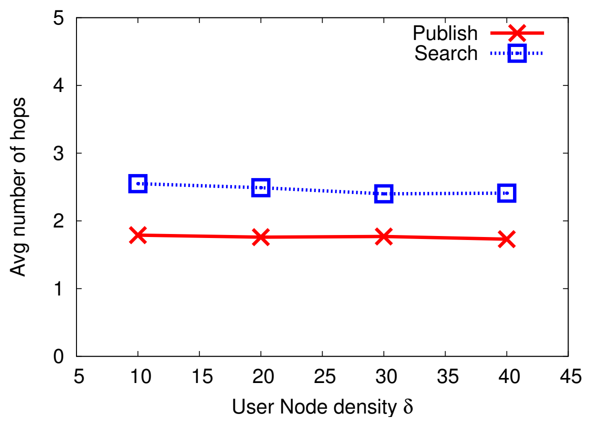

Figure 3: Average number of message propagations (hops) to discover SNs for publishing or searching data as functions of the UN density.

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.

A. Publication and Search

As described in Section III, the publication and search processes are distributed among all types of peers that participate in the DGT. For all simulated scenarios, due to DGT properties, all lookups were successful. Fig. 3 shows that the average number of message propagations (hops) to discover SNs for publishing or searching data as functions of δ. One should observe that these average number of hops are almost constant with respect to the density of UN. We remark that RDS nodes are initially aware of few SNs, but when their knowledge increases over a threshold, they stop performing storage node lookups. The average number of publication hops is always less than the average number of lookup hops. The reason is 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 angle α.

Fig. 4 illustrates the average number of exchanged messages for publication and search as functions of δ. The same considerations carried out in Fig. 3 for the average number of hops can be carried out in this case as well.

B. SN Disconnections

Fig. 5 shows that the resource availability, as a function of time, for the first scenario where there is a burst of SN

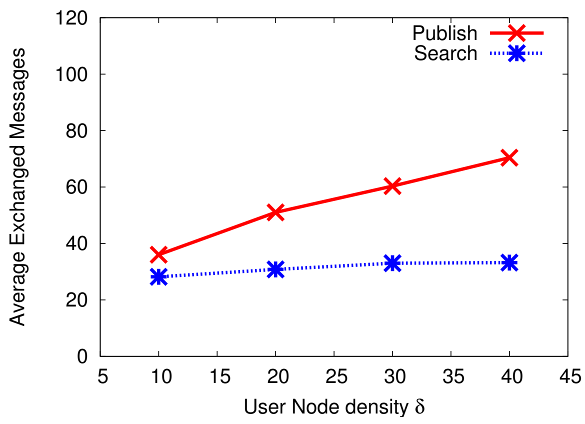

Figure 4: Average number of exchanged messages for publication and search as functions of the UN density.

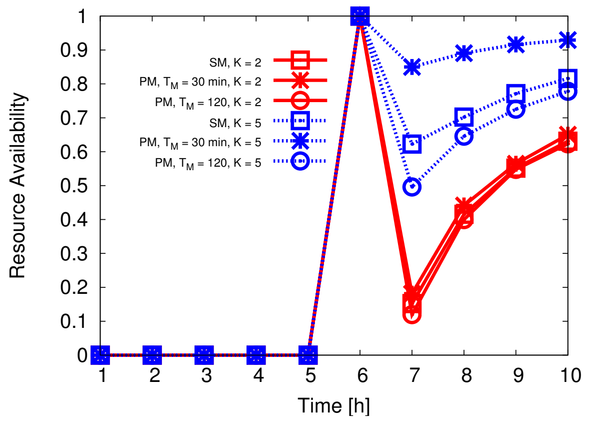

Figure 5: Resource availability, as a function of time, in the first scenario with a burst of SN disconnections. Both SM and SM (with different values of T_{M}) are considered.

disconnections. Either SM or SM (with different values of T_{M}) are considered. Note that only one curve is associated with SM, since 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, since the maintenance process allows to introduce new fragments in substitution of the lost ones. If $K = 2$ is considered, all strategies have less or more the same performance. On the other hand, if the publication overhead is larger, e.g., $K = 5$ 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 is the best choice if the maintenance is sufficiently frequent, whereas SM has to be preferred if the period of maintenance increases. 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 Fig. 6, the average occupied storage space per node is shown, as a function of time, with SM or PM (with $T_{\mathrm{M}} = 30$ min) and $K = 5$ (from Fig. 5 we have observed

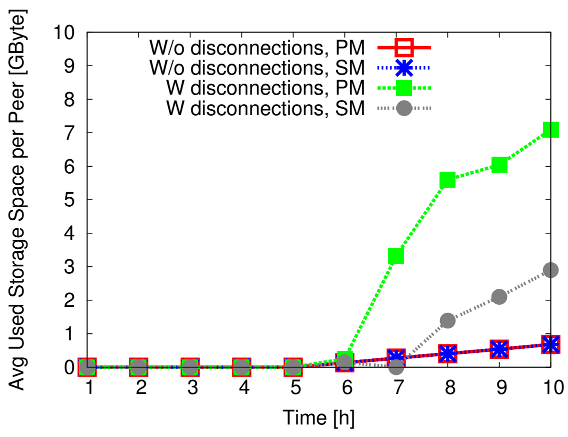

Figure 6: Average occupied storage space per node, as a function of time, with SM or PM (with $T_{\mathrm{M}} = 30$ min) and $K = 5$. The performance in the presence of the burst of SN disconnections is compared to that without disconnections.

that with $K = 2$ no performance variation can be observed). One can observe that in the absence of disconnections the maintenance strategy has no impact, since it can be shown that the same resource availability can be achieved. 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. Obviously, the larger is the number of fragments in the network, and therefore the occupied storage space, the higher is the resource availability. This can be verified by directly compared Fig. 6 with Fig. 5.

In Fig. 7, the resource availability is shown, as a function of time, in the second scenario with SN disconnections over a long period of time. Either SM or PM (with $T_{\mathrm{M}} = 1$ h) are considered. In this case as well, the maintenance processes preserve the resource availability to the maximum value (i.e., 1). In the presence of SM, however, the resource availability 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.

Fig. 8 shows the resource availability, as a function of time, in the third scenario with continuous SN disconnections and reconnections, comparing SM and PM (with $T_{\mathrm{M}} = 1$ h). One should observe that with PM the resource availability is not affected by such a churn process. SM, instead, is not sufficient to preserve resource availability to its maximum value. The reason is that if resources cannot be retrieved, due to SN disconnections, new fragments are not inserted in the system.

V. CONCLUDING REMARKS

In this paper, we have proposed a joint P2P/NC architecture for the management of large amount of information in smart

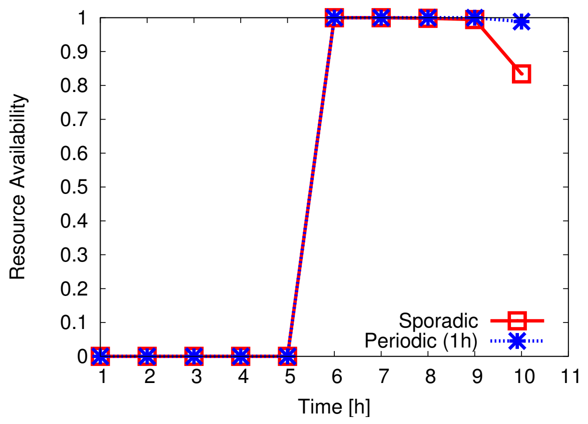

Figure 7: Resource availability, as a function of time, in the second scenario with SN disconnections over a long period of time. Both SM and PM (with $T_{\mathrm{M}} = 1$ h) are considered.

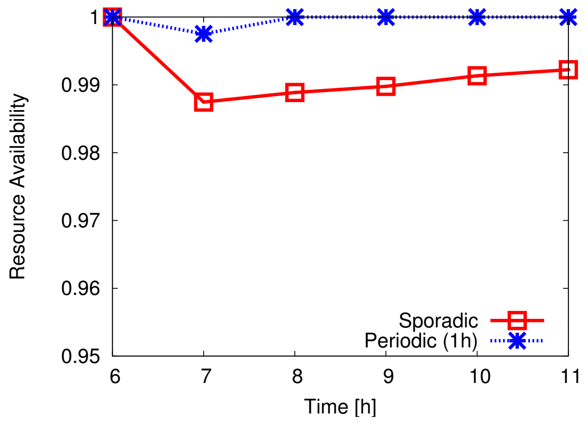

Figure 8: Resource availability, as a function of time, in the third scenario with continuous SN disconnections and reconnections, comparing SM and PM (with $T_{\mathrm{M}} = 1$ h).

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 two resource maintenance strategies, either periodic or sporadic, to overcome data losses.

Simulations of dynamic scenarios in a realistic environment (the city of Parma) have shown that the system is robust against storage node disconnections. Even in extreme conditions (e.g., 80% of storage nodes disconnected), maintenance strategies guarantee high resource availability. In particular, we have shown that PM performs slightly better than SM if maintenance is sufficiently frequent, at the expense of much higher storage space consumption. On the other hand, SM is effective only with popular resources, since regeneration is performed only after a successful download.

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. To this end, we are going to use the Sip2Peer middleware [23]—currently gaining momentum within the P2P community—to perform realistic experiments.

References

[1] Steventon, A., and Wright, S. (eds), *Intelligent spaces: The application of pervasive ICT*, London, Springer 2006.
[2] California Institute for Smart Communities, *Ten Steps to Becoming a Smart Community*, 2001.
[3] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. O. Wainwright, and K. Ramachandran, *Network coding for distributed storage systems*, IEEE Trans. Inform. Theory, vol. 56, no. 9, pp. 4539–4551, September 2010.
[4] J. Santa, A. Moragon, A. F. Gomez-Skarmeta, *Experimental evaluation of a novel vehicular communication paradigm based on cellular networks*, IEEE Intelligent Vehicles Symposium, Eindhoven, Netherlands, June 2008.
[5] R. Jedrzej, B. Scheuermann, M. Koegel, M. Mauve, *PeerTIS: a peer-to-peer traffic information system*. International Conference on Mobile Computing and Networking (MobiCom), Beijing, China, 2009.
[6] IBM, SmarterCity Initiative, *www.ibm.com/thesmartercity*, last access 16 September 2011.
[7] M. Amoretti, *A Survey of Peer-to-Peer Overlay Schemes: Effectiveness, Efficiency and Security*, Recent Patents on Computer Science, pp. 195-213, Vol. 2, Issue 3, ISSN 1874-4796, Ed. Bentham Science, November 2009.
[8] F. Oggier, A. Datta, *Byzantine Fault Tolerance of Regenerating Codes* IEEE Int.'l Conference on Peer-to-Peer Computing, Kyoto, Japan, August 2011.
[9] W. Rao, R. Vitenberg, S. Tarkoma *Towards Optimal Keyword-based Content Dissemination in DHT-based P2P Networks* IEEE Int.'l Conference on Peer-to-Peer Computing, Kyoto, Japan, August 2011.
[10] G. Mega, A. Montresor, G. P. Picco, *Efficient Dissemination in Decentralized Social Networks*, IEEE Int.'l Conference on Peer-to-Peer Computing, Kyoto, Japan, August 2011.
[11] A. Kovacevic, N. Liebau, R. Steinmetz, *Globase.KOM - A P2P Overlay for Fully Retrievable Location-based Search*, IEEE Int.'l Conference on Peer-to-Peer Computing, Galway, Ireland, September 2007.
[12] G. Montassier, T. Cholez, G. Doyen, R. Khatoun, I. Chrisment, O. Festor, *Content Pollution Quantification in Large P2P networks: a Measurement Study on KAD*, IEEE Int.'l Conference on Peer-to-Peer Computing, Kyoto, Japan, August 2011.
[13] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, *Network information flow*, IEEE Trans. Inform. Theory, vol. 46, no. 4, pp. 1204–1216, July 2000.
[14] M. Martalò, M. Picone, R. Bussandri, M. Amoretti, *A Pratical Network Coding Approach for Peer-to-Peer Distributed Storage*, Proc. of NetCod 2010, The 2010 IEEE International Symposium on Network Coding, Toronto, Canada, June 2010.
[15] C. Gkantsidis and J. Miller and P. Rodriguez, *Comprehensive View of a Live Network Coding P2P System*. Proc. ACM SIGCOMM Internet Measurement Conference (IMC '06), Rio de Janeiro, Brazil, 2006.
[16] M. Picone, M. Amoretti, F. Zanichelli, *Proactive Neighbor Localization Based on Distributed Geographic Table*, International Journal of Pervasive Computing and Communications, Vol.7, No.3, pp.240-263, October 2011, ISSN 1742-7371, Ed. Emerald.
[17] M. Picone, M. Amoretti, F. Zanichelli, *Evaluating the Robustness of the DGT Approach for Smartphone-based Vehicular Networks*, 5th IEEE Workshop On User MObility and VEhicular Networks, Bonn, Germany, 4-7 October 2011.
[18] I. Seskar, S. Marie, J. Holtzman, J. Wasserman, *Rate of location area updates in cellular systems*, IEEE Vehicular Technology Conference (VTC'92), Denver, Colorado, USA, May 1992.
[19] C. Fragouli and E. Soljanin, *Network Coding Fundamentals*, Hanover, MA, USA: Now Publisher Foundations and Trends in Networking, 2007.
[20] C. Fragouli and E. Soljanin, *Network Coding Applications*, Hanover, MA, USA: Now Publisher Foundations and Trends in Networking, 2007.
[21] M. Amoretti, M. Agosti, F. Zanichelli, *DEUS: a Discrete Event Universal Simulator*, 2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009), Roma, Italy, March 2009.
[22] Distributed System Group, *Smart City videos*, http://dsg.ce.unipr.it/d4v
[23] Distributed Systems Group, *Sip2Peer project home page*, http://code.google.com/p/sip2peer/.