Cloud Storage for Small Cell Networks

Ejder Baştuğ*, Jean-Louis Guénégo*, and Mérouane Debbah*
* Alcatel-Lucent Chair - SUPÉLEC, Gif-sur-Yvette, France
{ejder.bastug, jean-louis.guenego, merouane.debbah}@supelec.fr

***Abstract*—The Massive dense deployment of Small Cell Networks (SCNs) is a promising way of increasing capacity. Interestingly, by having such a huge amount of user devices as well as small cells deployed in indoor or outdoor areas, one can take benefit of such distributed network for storage purposes. Hence, Depending on the deployed scenario, these storage units can also be used for content caching to relieve the backhaul constraints and increase the peak rate. In this work, by extending concepts of cloud storage to SCNs, we discuss the theoretical challenges in order to embed small cells with storage capabilities. We also briefly introduce Open Cloud Protocol (OCP) as a unified software storage framework.**

***Index Terms*—cloud storage, private cloud, content caching, distributed storage, Small Cell networks, open cloud protocol.**

I. Introduction

In the years of social networks, high demand of rich content streaming by the users and/or machines is pushing the capacity of the legacy mobile wireless networks up to the limits [1]. Beside advances in legacy macro cell networks (MCNs), massively dense deployment of small cell networks (SCNs) has a particular interest to overcome this issue [2]. Moreover, the market forecasts expect almost fully SCNs scenarios in the near future [3].

Although their attractiveness, several challenges still exist. For instance, as the cell density increases, these low-cost, energy-efficient, self-configurable SCNs become more dependent to backhaul communications, due to their lower processing ability compared to MCNs. Assuming limited backhaul capacity, even using fiber-optic cables, connecting all small cells to the network is expensive. Therefore, by proposing SCNs as a solution, one should also take into account the capacity of the backhaul. One novel approach for this is to put high storage units to small cells and using cheap low-rate backhauls, rather than high-rate expensive alternatives [4].

In this work, we extend the idea of [4] by introducing concept of cloud storage on SCNs, and discuss possible scenarios. We then give theoretical challenges from the point of information theory, in the context of distributed storage. We present Open Cloud Protocol (OCP) as a unified software framework, and we finally conclude.

II. Scenarios

According to National Institute of Standards and Technology (NIST) definition of cloud computing [5], we can build up many scenarios depending on the deployment strategy (see Fig. 1).

A. Private cloud

It is a type of cloud where a single organization serves multiple customers in the network. In our case, this is typically a network operator that uses resources of user devices (i.e. mobile phone, laptop), small cells, data center or any combination of them as a distributed storage node. The operator can use this distributed storage for content streaming or persistent storage purposes. The work [4], [6] can be assumed as a private cloud, where fixed number of small cells are used for the content streaming.

B. Community cloud

This is the scenario where specific types of consumers in the community make a cloud for their missions. For instance, any user in the mobile network can socially share its mobile storage with other users in order to join this cloud.

C. Public cloud

The cloud is open to use and it might be owned, managed, and operated by a business, academic or government organization or combination of them. The local information of a city with augmented reality can be stored in the small cells or in other type of nodes, and the public users can take benefit of this service.

D. Hybrid cloud

Any combination of the clouds above can also be a cloud. In some sense, cloud of clouds might be called as hybrid cloud.

III. Challenges

By using the notion of information theory, we can describe possible challenges into four categories: 1) code design, 2) secrecy, 3) distributed storage allocation, and 4) proactive scheduling. Hereafter, we describe them step by step. Note that the metrics in the design can change depending on the small cell and backhaul deployment strategies. For example, we might have different metrics with different variability for small cells deployed in indoor, outdoor, hot-spot or rural areas.

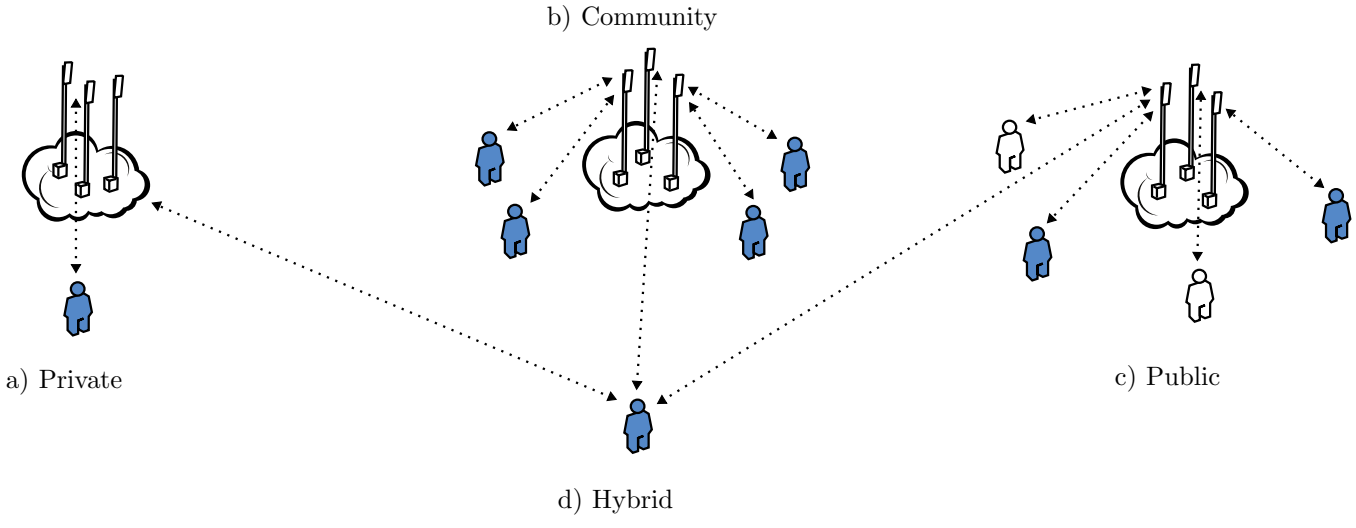

Figure 1: Possible scenarios on SCNs.

A. Code design

When we want to store the information to a distributed medium, some of the nodes may be in fail and/or they may not be available. In order to be able to overcome these failure events, and be able to obtain this distributed information later on in a reliable way, we might need to introduce some redundancy. The most redundant but reliable case would be to store that information between all the nodes, but in this case, the amount of the traffic overhead as well as storage efficiency might be non-desirable. Taking into account these constraints, we need to design the codes to distribute the information among the network, and we need to be able to repair and/or decode it under the given performance.

For the state of art, maximum distance separable (MDS) codes offer such a reliability by encoding k packets of the information into the n packets, where $n > k$. Then, we can sufficiently recover the original data. In the context of the distributed storage, n encoded packets can be stored in n nodes distributed in the network. After that, when a node fails, current nodes or incoming nodes can start a repair process in order to keep the constant reliability. The repair process is categorized as functional repair, exact repair and exact repair for the systematic part [7]. The functional repair has been studied theoretically in [8], but it has some practical implications due to the continuous update of the repairing and decoding functions. Indeed, such an operation may not be desirable, since an eavesdropper can detect the functionality that decreases the confidentiality of the private data. On the other hand, the exact repair problem is much harder and yet open in general. Significant amount of work has done in [8] and [7] according to the different performance metrics. The most important metrics are "repair bandwidth" and "storage efficiency". For the solution these extreme cases, exact minimum storage regenerating (MSR) codes [9] and exact minimum storage regenerating (MBR) codes [10], [11] have been proposed.

B. Secrecy

In the distributed scenario, an eavesdropper can actively read the data stored in a subset of storage nodes, or it may passively access the information generated during encoding, repairing and decoding processes. For such cases, the distributed storage system should be able to satisfy desired secrecy rate to protect the information.

Previously, [12], [13] have studied a passive eavesdropper case, where it reads the data stored in a subset of $l < k$ storage nodes, and it gets the information from $l' < l$ nodes during the functional repair process. The work [12] characterized the upper bound on the number of message symbols that we can store secretly, and showed that the MSR code can achieve the secrecy under an appropriate setting. On the other hand, [13] extended the idea and gave an explicit construction way for both MSR and MBR codes. The results of these works are built on undesired functional repair. Although, they are the first step towards the characterization of the secrecy capacity in the distributed storage. In complicated scenarios, like considering an open distributed storage system where the information of intended node must be kept secret to others, existing codes cannot be applied. Any information leakage from the intended node may result with detection of the deterministic encoding function from other nodes. Hence, one may suggest a secret key communication on top of the regenerating codes.

C. Distributed storage allocation

Consider a network where the nodes are not fully connected to each other, and the links between the nodes are capacity-limited. Once a node requests its distributed data

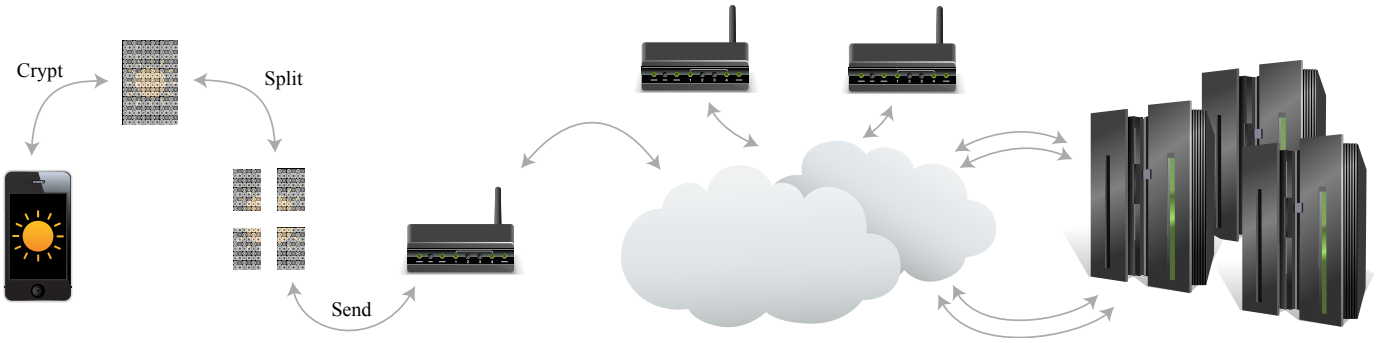

Figure 2: Storing and retrieving data in OCP.

from another node in the network, there might be a single or several links from the source to the destination. In such a case, the best option may not be the direct link -or the shortest path- due to the capacity constraints and the random failure probability, where subset of indirect paths with some caching nodes may achieve optimal information flow. Therefore, the distributed storage allocation problem arises on how to pick the best paths and how to allocate the information on the temporary nodes during the flow.

In [4], [6], the problem has been studied assuming that a central macro base station is sending information to its connected user, where a fixed amount of small-cells are in charge of caching the information to help the macro base station. The information cached inside the high storage capacity small cells is made of files with a given popularity distribution, and the links from the user to each small cell has a finite transmission rate. Under this setting, the optimal allocation to minimize the delay with respect to users and files has been found both for the coded and uncoded files. On the other hand, for the capture of the bursty and random behavior of the small cell availability, the works [14], [15] have modeled the link between the user and small cell as an erasure channel, i.e. the link is either switched on or switched off with some probability, and the statics is known to the allocation controller. More specifically, [14] supposes that only a random set of the small cells are accessible, while [15] models each link as the Bernoulli process with a given erasure probability.

D. Proactive scheduling

We have described above the distributed storage allocation problem for a given time instance assuming that the location of the nodes is fixed. However, in practice, some nodes may have mobility and make requests dynamically in different locations across time instances. If mobility and request pattern can be learned by the system [16] in a relatively long time scale (seasons, weeks, days, hours) and/or in a specific event, then, we can design our caching and transmission mechanisms more efficiently. Such a caching and scheduling would offer a great flexibility in the network.

A recent framework [17], [18] significantly reduces the outage capacity by predicting nodes' requests and anticipating the data transmission prior to the actual requests compared to a non-proactive case. Nevertheless, the model is a simplified queue system with aggregate capacity and does not consider the storage problem.

IV. Open Cloud Protocol

In this section, we introduce OCP as a unified software framework to securely and reliably store data in the network [19]. More precisely, each OCP installed device in the network is called as an OCP agent. The idea is to enable any electronic device with a memory and broadband internet access to be a node of a distributed system. Desired user experience has to be achieved through the following benefits: low cost storage, persistency, security, and performance of storing and retrieving data. OCP is designed to allow an online storage market with many actors: consumers, providers and traders, it includes billing functionality. On the other hand, it targets to have a flexible working architecture starting from physical layer to the top, including ability of working for different deployment scenarios as described above. A rough visualization of storing and retrieving data in OCP is shown in Fig. 2.

A. Address topology

Every storage system has the concept of address, that allows it to know where an object can be stored inside its containers. Any distributed storage system has an address topology, which is a $triplet(F, U, R)$ with F representing the address format, U the set of all possible addresses, called the address universe, and R being the repartition rules that specify for each address which node is responsible for it and when. Any distributed hash table (DHT) has an address topology. For instance CHORD [20], Kademlia [21] and Tapestry [22] have a single ring address topology. The DHT content addressable network (CAN) has a multi-dimensional box address topology [23]. The particularity of OCP is to have a multi-ring address topology.

B. Multiple Ring address topology

The multi-ring address topology is a $triplet(F, U, R)$ with F defined as a one field address, U as a finite set,

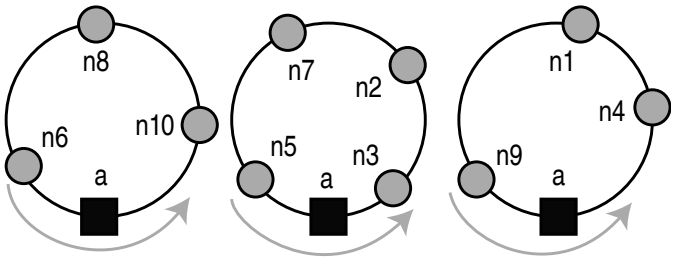

Figure 3: Multiple Ring address topology.

and R as follows. Let us consider N rings, each ring being a U representation as a counter-clockwise directed ring. To each node, is assigned a ring r and an element e of the ring which has an address format. The $couple(r, e)$ is called node id. This gives to each node a position on one ring. A node B is successor of a node A if and only if they are on the same ring, and there is no node between them on this ring, and node B is after A in the specified ring direction. A node is responsible for all object with an address between it and its successor. The set of addresses under the responsibility of a node is called *node responsibility area*. This kind of topology allows to easily manage redundancy, and the action to perform on node disappearing events. An example of this topology is shown in Fig. 3. In the figure, we can see a three ring address topology with 10 nodes. The black square represents an object located at the address a. According to the repartition rules R, a is under the responsibility of nodes $n6$, $n5$ and $n9$. The *responsibility area* of these nodes are indicated with the arrows on the figure.

C. Immutable objects

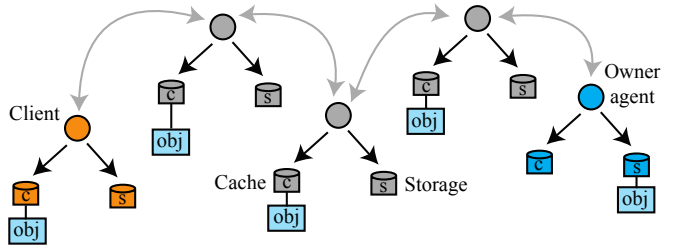

Figure 4: Cache on the road.

OCP particularity is to extensively use immutable objects and directed acyclic graph (DAG) like GIT versioning system. An immutable object is an object that cannot be modified. This is done by linking its content to its address with a relation such as $address = hash(content)$. An immutable object cannot be updated while keeping its address. So it becomes difficult for a node to modify an object. Another advantage of using immutable objects is that they can be cached on any node. This allows a mechanism that we call as *caching on the road*: when retrieving an object, the object can be cached on all devices between the client and the responsible node (see Fig. 4). When a node receives an object, it stores in its cache if it is not responsible of it, else it stores in its persistent storage area. Cache policies remove the object from the cache when needed. Moreover, if the user client has mobility, the caching on the road mechanism will have an effect to keep data geographically close to the client.

D. Mutable objects

OCP, however, uses mutable objects for connection objects. A connection object is an object stored at a specific address that can be retrieved by a user without specific information. For instance, OCP uses public user object. For a given user with a login, the public user object is located at $address = hash(login)$. OCP uses as well private user object. For a given user with a login and a password, the private user object is located at $address = hash(f(login, password))$. The private user object is a password crypted object containing a pointer to the root of the user container, represented and stored under a DAG. OCP mutable objects cannot be cached.

E. User objects

In OCP, all objects belong to a specified user. This allows the agent to publish billing report. All immutable objects are encrypted with a user secret key located in the private user object. When a user stores a file, the file is splitted in many relatively small blocks that will be spread around the agents.

F. Storing data

To store data on an OCP network, the user starts an OCP agent acting as a client. It creates an account or uses its existing one. The user can login with an authentication challenge. The login step consists to determine the address of its public and private user objects. Storing a data (file, stream, etc.) consists of encrypting the data then splitting the encrypted data into blocks and adding redundancy through an erasure code, and finally sending the blocks on the OCP multiple ring address topology DHT as immutable objects organized in DAG (see Fig. 2). Caching on the road mechanism is used.

G. Retrieving data

To retrieve data, the connected user retrieves all the addresses of immutable objects via the DAG root address stored in the private user object. Data is rebuilt from different blocks using erasure code, and then decrypted in order to be manipulated by the user.

H. Cache and persistent storage

OCP agent has two kinds of containers: one for storing data the agent is responsible for and another one for cached immutable objects. As a consequence, an OCP network can be composed of cache oriented agent and persistent storage data agent. This helps to have a storage network with independent persistence and performance aspects.

V. Conclusion

By increasing the capacity of wireless networks with SCNs, we have shown that we can take benefit of such a huge network for distributed storage. We have called this concept as cloud storage on SCNs. Giving these possible scenarios and existing technical challenges, we believe that there will be extensive research and engineering work in this direction, in the near future.

VI. Acknowledgment

This research has been supported by the ERC Starting Grant 305123 MORE (Advanced Mathematical Tools for Complex Network Engineering), and Alcatel-Lucent, within the Alcatel-Lucent Chair in Flexible Radio, Supélec.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2011–2016," *White Paper, [Online] http://goo.gl/lLkTl*, 2012.

[2] J. Hoydis, M. Kobayashi, and M. Debbah, "Green small-cell networks," *IEEE Vehicular Technology Magazine*, vol. 6(1), pp. 37–43, 2011.

[3] Small Cell Forum, "Small cells to make up almost 90% of all base stations by 2016," *[Online] http://goo.gl/qFkpO*, 2012.

[4] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," *arXiv preprint: 1109.4179*, 2011.

[5] P. Mell and T. Grance, "The nist definition of cloud computing," *National Institute of Standards and Technology*, 2011.

[6] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *arXiv preprint: 1204.1595*, 2012.

[7] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, 2011.

[8] A. G. Dimakis, B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *Information Theory, IEEE Transactions on*, vol. 56, no. 9, pp. 4539 –4551, 2010.

[9] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Exact regenerating codes for distributed storage," in *Communication, Control, and Computing (Allerton 2009). 47th Annual Allerton Conference on*, pp. 1243 –1249, 2009.

[10] C. Suh and K. Ramchandran, "Exact regeneration codes for distributed storage repair using interference alignment," *arXiv preprint: 1001.0107*, 2010.

[11] Y. Wu and A. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Information Theory, (ISIT 2009). IEEE International Symposium on*, vol. 4, pp. 2276 –2280, 2009.

[12] S. Pawar, S. Y. E. Rouayheb, and K. Ramchandran, "On secure distributed data storage under repair dynamics," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, p. 2543–2547, 2010.

[13] N. B. Shah, K. V. Rashmi, and P. V. Kumar, "Information-theoretically secure regenerating codes for distributed storage," in *GLOBECOM, Auisting, Texas*, pp. 1–5, 2011.

[14] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocation for high reliability," in *Communications (ICC), 2010 IEEE International Conference on*, pp. 1 –6, 2010.

[15] V. Ntranos, G. Caire, and A. G. Dimakis, "Allocations for heterogenous distributed storage," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pp. 2761 – 2765, 2012.

[16] V. Etter, M. Kafsi, and E. Kazemi, "Been There, Done That: What Your Mobility Traces Reveal about Your Behavior," in *Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf. on Pervasive Computing*, 2012.

[17] H. E. Gamal, J. Tadrous, and A. Eryilmaz, "Proactive resource allocation: Turning predictable behavior into spectral gain," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pp. 427 – 434, 2010.

[18] J. Tadrous, A. Eryilmaz, and H. E. Gamal, "Proactive resource allocation: Harnessing the diversity and multicast gains," *arXiv preprint: 1110.4703*, 2011.

[19] "Open Cloud Protocol," *[Online] http://www.flexible-radio.com/ocp*, 2012.

[20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 149–160, Aug. 2001.

[21] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, (London, UK, UK), pp. 53–65, Springer-Verlag, 2002.

[22] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," vol. 22, pp. 41– 53, 2004.

[23] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 161–172, Aug. 2001.