A CCN-Based Social Network Application Optimising Network Proximity

Bertrand Mathieu, Patrick Truong
Orange, Lannion, France
{bertrand2.mathieu, patrick.truong}@orange.com

***Abstract*—Online Social Networking (OSN) applications attracted millions of people in few years and are considered as the success story of current Internet applications. However, how they work is unclear for both end-users and researchers, since the developers keep the system architecture secret and use encryption mechanisms. In this paper, we present the main outcomes of our analysis of one of the most well known OSNs, Twitter, focusing on the networking behaviour, the involved entities, their location, etc. Detecting that the current network behaviour of those applications is not in line with current end-users behaviours, where locality is important, we propose to adapt the OSNs to a Content-Centric Networking (CCN) approach, which could improve the delivery and reduce the network load and server load. Indeed, CCN is a new networking paradigm focusing on interest based requests of content itself independently on location and so has features very close to OSNs. This paper then introduces our proposal for a CCN-based architecture for OSN applications, with a naming and routing proposal. We also present the demonstrator we have implemented as proof-of-concept and the testbed we set up for functional testing purposes.**

I. Introduction

Internet usage has rapidly evolved over the last decade, moving beyond simple one-to-one connections toward more complex interconnections between hosts. Content consumption is changing, users migrating away from web browsing to get information. Where information was previously stored sparsely and locally in some specific network locations, it can now be massively disseminated and even replicated in numerous different platforms, so that requesting information can be performed from a large range of sources. The breakthrough development of peer-to-peer networking connectivities has first considerably contributed to this shift in content consumption, before giving way for the trending popularity of Online Social Networks (OSN). Social networking services are distributed platforms through which people sharing the same interests, activities, background and so on can interact. The most prominent OSNs are Facebook and Twitter.

To better meet and scale with the challenging networking requirements related to the breakdown in content consumption, important research efforts have thus been fostered last years to envision clean foundations for the Internet architecture. While the IP protocol has always been shaped to be agnostic at best with the ever-growing amount of network data or services that often requires real time delivery constraints, the end-to-end principle on which Internet was designed is no longer usable to face up to the current emergence of intensive content-consuming applications, taking part to the big data explosion. As a consequence, Information-Centric Networking (ICN) has recently been promoted as a promising paradigm to tackle the shortcomings of the IP host-centric model. Several ICN architectures [3] have been proposed with a persistent and location-independent naming scheme for content objects so that routing and forwarding are done directly on content, and not anymore between hosts.

As the ICN paradigm promotes the interconnection of content objects rather than the hosts where they are located, interdependencies between network locations are completely redefined to be more dependent on users' interest. A recent study [2] shows how following discussions on Twitter can be used to predict popularity of content on Internet, influenced by the way OSNs' users can turn information into a buzz. Leveraged by OSNs and various other social media websites (such as YouTube), content is not only easy to find, but it is also easy to generate, share and be disseminated in several parts of Internet. Despite its huge success, OSNs have not being, to the best of our knowledge, largely analysed from the networking point of view. The authors of [10] have done preliminary work detailing what happens when users get access to their Facebook home pages. They tried to answer questions of how many and in which order content objects are retrieved, which and how many servers are involved, how many TCP streams used to satisfy users requirements. Based on the paper [7] which provides promising simulation results for the performance of using a CCN architecture [5] as content delivery for OSNs such as Twitter, we perform further investigations into the network traffic to identify that OSN architectures are not aligned with the end-users' behaviour. We then consider the content naming and routing issues when using CCN to convey the networking interactions of a Twitter-like social application.

The rest of the paper is organized as follows. In section II, we present an analysis of Twitter to highlight the networking call flows. We then propose our CCN-based social network application in Section III. Section IV introduces the demonstrator we have developed and the testbed we have set up to evaluate it. Finally, we conclude our paper in Section V.

II. Analysis of Twitter Networking Behaviour

In this section, we analyse the networking behaviour of the Twitter application. Our research consisted of two steps, capturing packets of the exchanged traffic and analyzing the results. The traffic analysis was done by using a web

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

browser (Mozilla Firefox) with an ad-on called *Firebug* (a web development toolkit that facilitates the debugging, editing, and monitoring of any website's CSS, HTML, DOM, XHR, JavaScript and provides also other web development tools) and the Wireshark packet capturing program for more details. Four Twitter accounts attached to few/many memberships, including also active-accounts (active-account means that the user frequently sends tweets to their followers, i.e. France24, TF1) to carry out our research. To determine the location of servers bases on their IP addresses, we used *IP2Location.com* (a non-intrusive geo IP solution to help identifying visitor's geographical location). All data were collected through an ADSL connection of the Orange network with all accounts. The operation was started by running the browser and clearing the history and caches before connecting to the websites. Full packets from Twitter accounts were collected gradually for analysis, but unfortunately, Twitter only proposes HTTPS sessions for all the connections. This latter caused us a problem to view the sent/received data from/to servers which were encrypted. However, the Firebug plug-in gave us some details about the requests and data exchanged between both involved endpoints. Indeed, the plug-in gets data before its encryption for transmission and after decrypting them before display to the user.

Figure 1 depicts the involved entities for the Twitter service.

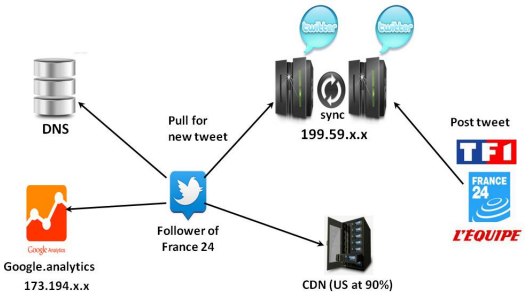

Fig. 1. Involved entities in the Twitter Service.

It was found that for a period of seven hours' duration, the user conducted on average 13 DNS requests per hour to get the IP address of one Twitter server. The received IP address always referred to one server located in US (199.59.x.x). Our client then connected to this server. It was not always connected to the same server, the client frequently changed after each DNS request, but the server IP address was always in the subnet 199.59.x.x. This gives us a hint that Twitter works under a load balancing system with synchronized and redundant servers, deployed in US. Once the connection is established, a counter of 10 seconds is started. If the counter reaches its limit and no action (e.g. sending tweet, updating profile, etc.) occurs during the interval, the connection automatically terminates. Otherwise, exchanging data allows to keep the connection alive until both endpoints have finished their transfers.

Our investigation shows that for one user, the data related to tweets (sent and received) are always in relation with the Twitter servers located in US (subnet 199.59.x.x). Few other connections are established with other servers, mainly to retrieve the profile image or some ads. These connections are established toward CDN servers located in (US, FR, DE, UK). From our experiment, a high percentage of those connections are with the CDN servers located in US (68.232.x.x), but very few with those in Europe. Objects already retrieved are to be saved in the browser cache, so that the next requests for the same objects will be retrieved locally and rapidly. Finally, Twitter provides also a connection with Google-analytics in order to have some information about Twitter users, such as statistics about tweet, re-tweet, and other metrics.

Each time a user posts a tweet, connections toward Twitter US servers are established to upload users data if there is none already in progress. A follower can retrieve published tweets (from people she/he follows) or update her/his profile via two ways. First, like for posting a tweet; if there is already a connection in progress, this one will handle data transfer from server to the user. Otherwise, if there is no established connection, the user will establish a connection toward the US Twitter server every 30s to request tweet publications time updates. The polling time remains fixed (30s) when there is activity from both sides (user or server), e.g.; the user/server send/receive data. If there is a long period of inactivity (no tweet sent or received by the end-user), the polling time follows an increasing automation process shown in Figure 2. It reaches its maximum time (1200 sec), after about 1 hour of inactivity. This polling time only increases when there is inactivity and as such, this behaviour is not detected for end-users following accounts that frequently send tweets. Indeed, our end-users following France24, TF1 (or simlar active accounts), frequently received tweet and stay with the 30s duration polling time.

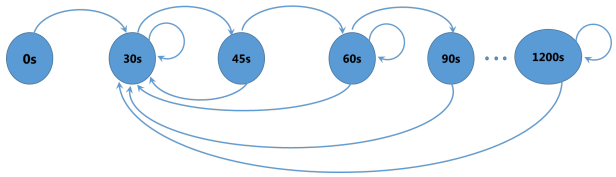

Fig. 2. Twitter polling time evolution

At the reception of any action produced by the end-user or the server, the counter is reset to the initial state (30s).

We also analyzed the involved Twitter servers for end-users located in a several countries (Germany, Sweden, Australia, Poland, USA, United Kingdom, Japan, Switzerland). It was done using the PlanetLab network [1] (which gives us access to remote machines, located at various places in the world). For each country, we performed tests with three nodes. The results show that the users always communicate with the same family of IP addresses (199.59.x.x), i.e.; the end-users are connected to the clusters of Twitter servers located in US, wherever they are located.

III. Proposal for a CCN-Based OSN Application

In this section, we introduce our proposal for a CCN-based solution acting as content delivery for OSN applications. First, we depict the OSN users' behaviour which is different to OSN networking behaviour before proposing our CCN-based approach for social networking applications. This encompasses the naming and routing solutions we propose as well as the application algorithms to improve the delivery.

A. OSN Users Behaviour vs OSN Networking Behaviour

Analysing the OSN end-users behaviour is done by several research teams, wishing to better understand the social relationships between users, their habits, their way of using and consuming OSNs, etc. The papers focus on the shared content itself and the social graph between users. From those papers, such as [9] [4], we can say that locality plays an important role in OSN applications. People are very frequently connected to other people that are in the same town, same region, in short in a close vicinity (e.g. tweets are distributed locally to local followers, users often send their tweets from the same location, etc.), except for very popular accounts (e.g. a Twitter account having millions of followers). Therefore, OSN messages which are exchanged between end-users are mainly destinated to local or close users. Having analyzed the OSN networking behaviour in the previous section, we can see that the end-users behaviour is not aligned with the OSN network behaviour, which always incurs transfer of messages toward remote centralised servers, while the real destinations of the messages are very close. The network architecture could then be optimised so as to better reflect the end-users behaviour.

B. Naming of the CCN-Based OSN Objects

Based on the analysis of the OSN networking behaviour and the end-users behaviour just presented, we propose a naming scheme depending on the end-users' characteristics. We define two types of end-users: the local ones and the non-local ones. The local end-users are the ones who have friends/followers only in the same region, area: they are then local. The non-local ones are the non-popular end-users, who have friends/followers very far (e.g.; a cousin in Australia, whereas she is in France), or the popular end-users who have friends/followers everywhere. Popular end-users, whose content is consumed worldwide, should have a different way of working than non-popular local end-users, whose content will be locally consumed. In the first case, the current networking behaviour of OSNs can be respected but for the latter, a CCN-optimised delivery is envisioned, taking into account appropriate routing strategies based on a specific content naming scheme. The proposed naming differentiates end-users, whether they produce contents which are locally consumed or not, and is as follows:

- For Non Local end-users, any related content is named as /Twitter/NonLocal/UserXXX/Tweet_YYY
- For Local end-users, the naming prefix is different: /Twitter/Local/UserAAA/Tweet_BBB

With such a difference in the first segments on the names, the CCN routers will be able to easily route the interest message toward the OSN servers (non local users) or towards the end-users (local users). We should then have different entries in the Forwarding Information Base (FIB) of the CCN routers. We also propose to name the tweets with a numbering, allowing end-users to know which tweet to request. Indeed, if an end-user has already retrieved the tweet #25 from the user "Bertrand", it knows that the next tweet to request is Tweet_26. It will then send an interest for "/Twitter/<Non>Local/Bertrand/Tweet_26". Depending whether Bertrand is local or not, the interest request will be forwarded differently.

For adopting this naming scheme, it is assumed that the end-user must declare at the beginning herself/himself as a local or non-local end-user, but the OSN application needs to determine in-service when and how to change her/his status (and therefore the naming) of the user, e.g. if the user becomes popular or if some of her/his friends move away and live in different countries. This can be done with an analysis of the social graph of the user and a reconfiguration announcement in the network for the adaptive naming change needs then to be defined for the user, and the history of past messages could be imported in the new profile for being remotely accessed. The user's followers can thus be informed by the status change by the application.

C. Routing of the CCN-Based OSN Objects

We advocate a semi-distributed system where local users will have an adapted way of working and popular or non local users will still have a behaviour similar to the current one. It enables OSN providers to have knowledge of its clients and possibly adapt some processing for very popular end-users. Furthermore, it can be an added value for them (and something they can monetize) for popularity measurements, targeted advertisement (e.g. announcement of a live of a popular singer that is followed by millions of people, etc.).

Local end-users represent a huge number of customers, and a large majority of them do not have many followers who are, for most of them, also located locally. Since there are many end-users in this situation, processing and routing tweets locally in an efficient way can lead to a large reduction of network traffic and processing load on the OSN servers. We then propose a distributed way of working for such local users.

Based on the distinction between non-local and local users, the CCN routers need then to be able to route interest messages toward the OSN server or directly toward local users. This is possible via the different prefix we use in the naming of content. Every content name with prefix "/Twitter/NonLocal/" will be forwarded toward the OSN server whereas content names "/Twitter/Local/UsersXXX/..." will be forwarded to the user XXX. The CCN routers should thus be aware of the users' locality as described in the following.

For the popular or non-local end-users, the Twitter server will be in charge of announcing the name prefix "/Twitter/NonLocal" in the network, using a routing protocol such as

OSPFN [8]. Each CCN router FIB contains then one entry for routing to the Twitter server all content requests related to non-local end-users. As the Twitter server can be far away in the network (e.g. in US and users can be in Croatia), the announcement message should have a huge TTL (Time-To-Live) value to be propagated worldwide. It can lead to a worldwide flooding network but since the Twitter server is stable, always on, and with a simple prefix, this announcement will be rare (just to inform it is still there) but there is no need for frequent updates.

When retrieving non-local users' tweets, all interests with the name prefix "/Twitter/NonLocal" should normally be forwarded to the Twitter server. However, due to the caching feature of CCN, the intermediate routers on the network path can have the requested content already cached (further to a previous request by another follower) and therefore can serve immediately the content, leading to a reduced load for the Twitter server as well as for the network bandwidth. In the same way, if several interests arrive almost simultaneously for one user, the CCN router will not forward the second interest (and the subsequent ones) but will just wait for the coming Data related to the first sent interest message and will forward this Data to all the requesting followers. For those popular end-users, we can then see the benefits of using the CCN architecture with its native caching and multicast features.

For the local end-users, we adopt a distributed architecture to alleviate the load on the Twitter server and to reduce the network load between the users' access networks and the Twitter server, as the majority of the traffic stays locally. As such, interest requests for local content are sent toward the local end-users themselves who need accordingly to announce in advance their name prefix in the network. The OSN application on the client side should then be adapted to reply to those interest requests, acting as a server.

Since it is not realistic to think that every router will keep knowledge of all local end-users in their FIB table (for scalability issues), we rather advocate for a dynamic and temporary configuration of the FIB tables. Indeed, the local end-users generally send few tweets (compared to non-local producers such as France24, TF1, etc.) and the tweets are not time-constraint. We then propose that they have sporadic reachability. When those end-users want to publish a tweet, they send a prefix announcement in the network with their contentName domain, so as to inform the CCN routers. The CCN routers will then populate their FIB accordingly and be able to route interests for such contentName toward those end-users. It is similar to what [6] proposed and called on-demand publishing. For the announcement, as for the Twitter server, the local end-users can announce their prefix name (e.g. "/Twitter/Local/Bertrand/") with the OSPFN protocol [8], but with a limited TTL value so as to reflect the locality of interest. This entry will then be present only in the FIB of CCN routers being at 1 or 2 (value to determine) hops from the end-user. Moreover, in order not to overload the FIB tables of the CCN routers, we propose to announce the local end-user domain just for a small period of time (e.g. 5 minutes or another value to determine to optimize the system). This end-user will then be reachable just for 5 minutes. Now, as followers regularly send interest messages for new tweet (e.g. every 30 seconds or a bit more in a progressive manner: see section II), this interest will certainly be in this time period and thus be able to be routed toward the end-user that has produced the tweet. However, some followers can be offline during this period, so to allow them to retrieve the last tweets, we propose that local end-users periodically announce their contentName domain for another period of 5 minutes (even if they have no new tweet to publish) in a regular manner (e.g. every 3 hours or another value to determine to optimize the system). The followers who are offline during the first period, will be able to get them in the next period (or even the one after), when they will be back online. The values proposed should be determined according: 1) to the number of simultaneous entries in the FIB table so as to have a scalable system and 2) to the online presence time of end-users, so as to guarantee that followers get the tweets not too late (e.g. not 3 days after). Between the 3-hours publication period, the local end-users will not be reachable and when a follower will try to pull for new tweet, since there is no related entry in the FIB, the CCN router will just discard this interest after a short time-out (as explained in [5]). To further offload this sporadic reachability, the OSN application can additionally be configured to prefetch and store in advance content of interest for the customer directly in the end-user terminals, at the most appropriate place (e.g. according to the followers' location) and at the most appropriate time (e.g. according to WIFI connection availability).

Figure 3 depicts the proposed way of working with one non-local publisher (France24), one local publisher end-user (Bertrand) who also follows France24, one remote user following just France24, and 2 others close to Bertrand following both Bertrand and France24. The figure shows the announcement phase as well as how to send/get tweets.

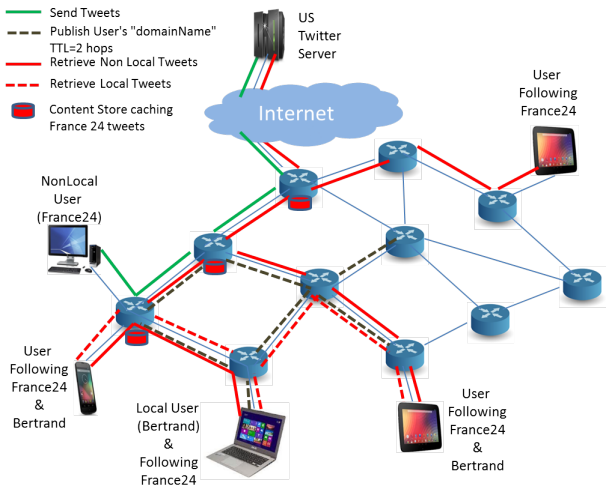

Fig. 3. CCN-based Twitter-like Application

D. Application Call Flows

In this section, we present the call flows defined to send/receive tweets from local and non-local end-users.

1) Publication of Tweets: On the Twitter server side, at startup, the server opens a connection listening on the prefix "/Twitter", so as to receive interests in sending tweets. If a user (let us say Bertrand) wants to send or post a tweet, he requests the server to send a file (named "SendTweet_Bertrand"), containing the number that the user must use to name the file containing the tweet to publish. At the reception of the data for this file, the end-user can then write his tweet and save it in the file named "Tweet_N". After having sent the file "SendTweet_Bertrand", the server prepares an *Interest* request for this content "Tweet_N". On the user side (Bertrand), the software is listening on the prefix "/Twitter/ForServer_Bertrand" and when it receives the interest, it replies with the data file named "Tweet_N" containing the tweet to publish. The Figure 4 depicts this call flow.

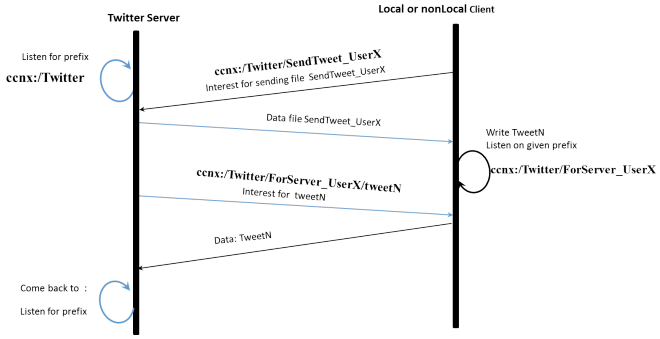

Fig. 4. Publication of a tweet

2) Reception of Non-Local Tweets: For users following non-local users, given that the tweets are served by the Twitter server, the process is simply similar to the current one done by Twitter. The user sends the *Interest* message related to the tweet "/Twitter/NonLocal/UserX/Tweet_N" to the Twitter server. The user knows what X is (the name of the followed user) and what N is (the number of the tweet just after the last one he has received). The *Interest* is routed in the network via the prefix ("/Twitter/NonLocal") announced by the Twitter server. When receiving this *Interest*, the Twitter server sends back the data related to the queried tweet (the server has already received this data during the tweet publication phase). Subsequent to this, the users send an *Interest* for the next tweet (N+1) and if it has not yet been published, the Twitter server will not reply. After the expiration time, the user will issue another *Interest* following the algorithm described in Section II. As soon as the tweet has been published by the owner, the server will then be able to reply to this *Interest*.The Figure 5 describes the retrieval of non-local tweets.

3) Reception of Local Tweets: For users following local users, the *Interest* message need now to be routed toward the local end-users themselves, who will then reply directly to the requester. As the *Interest* is not sent to the remote Twitter

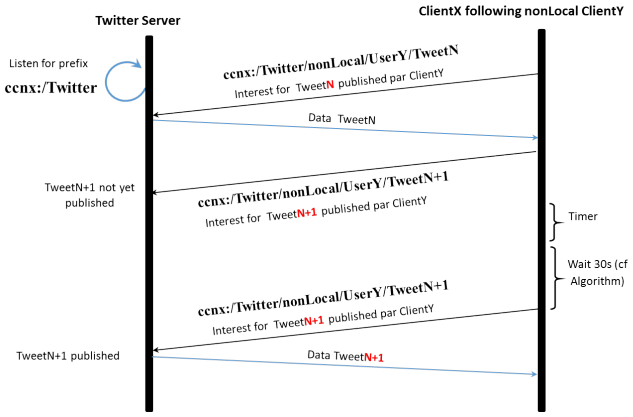

Fig. 5. Reception of a non-local tweet

server, this contributes to reduce the overall network load as well as the server load. To be reachable, the local user should have announced his name prefix in the network, as mentioned in Section III-C, just before publishing a tweet and regularly so that to be available to users who have missed the previous period of availability. The Figure 6 describes this process.

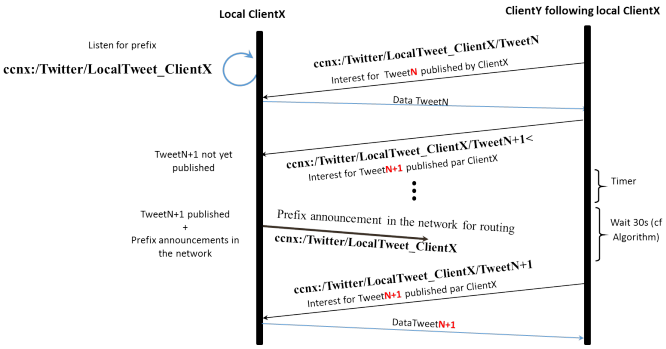

Fig. 6. Reception of a local tweet

IV. Testbed Implementation for Functional Tests

We have developed our CCN-based Twitter-like application using the CCNx library for the network stack and Java for the application logic and GUI related to the social application (Figure 7). The CCNx API is responsible for the basic service of CCN (FIB tables, PIT, sending Interest, Register in caches, etc). Java was used for its ease of programming, the GUI library and because CCNx offers a Java library.

For proving the feasibility of the approach, we have implemented a demonstrator for evaluation on a testbed hosted inside one Orange specific network dedicated to perform functional tests. The testbed includes different network regions to illustrate the locality-aware behaviour as identified in sectionIII-A. We set up a realistic networking environment with various network structures (access, aggregation, core and interconnection). The access network is in ADSL, FTTH or Ethernet. We use a Linux PC for the Twitter server, and the Twitter clients run on laptops. The testbed configuration is as follows:

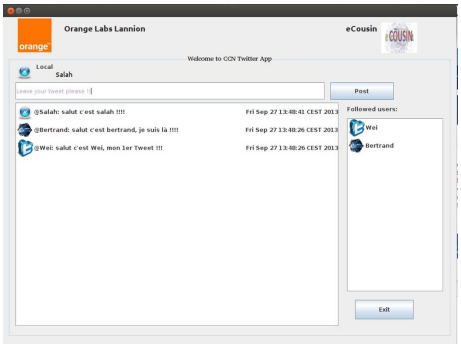

Fig. 7. Twitter-like application

- a Twitter server will be set up in a remote AS (similarly to the official Twitter server in US): region 4 n our testbed
- end-users in 3 different regions to show local (and non-local) activity
- CCN routers deployed in edge nodes

For lack of a existing content-centric network layer, i.e. whose forwarding decisions are solely based on named pieces of content (instead of IP addresses as it is the case for the current IP network layer), our CCN-based architecture for delivering content in Twitter is deployed as an overlay network. The CCN routers are hosted in Linux-based machines. The Figure 8 depicts the testbed we envision for this use-case.

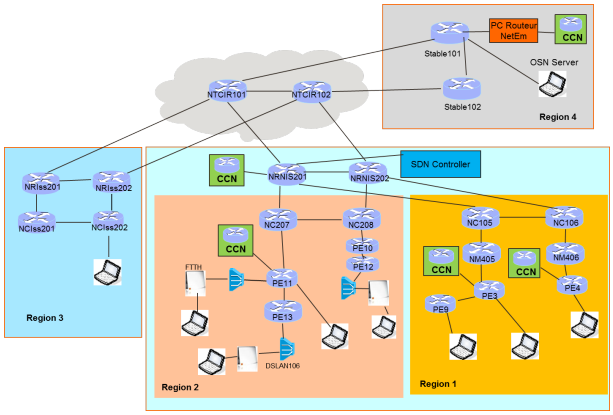

Fig. 8. Orange representative Testbed

As functional tests, we successfully verify that our locality-aware naming scheme is valid and that our content-centric routing approach based on this naming scheme can work for publication and retrieval of tweets as expected. However, as a next step, we need to evaluate the performance and efficiency of routing and caching, showing that processing and routing tweets locally can lead to a large reduction of network traffic and processing load on the OSN servers. The expected results should quantify the benefits of using CCN as a content transport for an OSN such as Twitter.

V. Conclusion and Future Work

In this paper, we have presented how one of the most well-known social network applications (Twitter) behave from the network point of view. We highlighted that their current networking design is not adapted to the current users' behaviour, where locality plays a major role in the social relationships. To overcome the current network limitations, we proposed to use the new CCN network paradigm, that presents similar features to OSN applications and can be well suited. We have presented our CCN-based OSN application, with the naming and routing issues and our adaptation to the publication and reception of tweets. The solution we have defined aims at reducing the network load (of advantage for network operators), while optimising the delivery of information (better Quality of Experience for end-users) and at reducing the server load (of advantage for the OSN provider). Furthermore, via the defined call flows, the OSN server is still aware of all its end-users, their relationships and the produced tweets. It can then save its business model, based on data mining, advertisements, etc. We have set up a tesbed in order to evaluate our solution and made initial functional tests to prove its feasibility. Our future work is evaluation performance so as to be able to identify the network gain and server gain using local-aware routing for local users. We also intend to add a video feature to our Twitter-like application so that end-users can share not only small texts but also videos. If such a use-case becomes a success story, our solution will even be more interesting since the network and server gains will be much more valuable.

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme ([FP7/2007-2013]) under grant agreement n°318398, project eCOUSIN.

References

[1] The planetlab network. http://www.planet-lab.org/.
[2] S. Agarwal and S. Agarwal. Social networks as internet barometers for optimizing content delivery networks. In *Advanced Networks and Telecommunication Systems (ANTS), 2009 IEEE 3rd International Symposium on*, pages 1–3, 2009.
[3] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, vol.50, July 2012.
[4] R. Cuevas, R. Gonzalez, A. Cuevas, and C. Guerrero. Understanding the locality effect in twitter: measurement and analysis. *Personal and Ubiquitous Computing*, pages 1–15, 2013.
[5] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. CoNEXT '09. ACM, 2009.
[6] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard. Voccn: voice-over content-centric networks. In *Proceedings of the 2009 workshop on Re-architecting the internet*, ReArch '09, pages 1–6, New York, NY, USA, 2009. ACM.
[7] B. Mathieu, P. Truong, W. You, and J.-F. Peltier. Information-centric networking: a natural design for social network applications. *IEEE Communications Magazine*, pages 44–51, 2012.
[8] L. Wang, M. A. Hoque, C. Yi, A. Alyyan, and B. Zhang. OSPFN: An OSPF Based Routing Protocol for Named Data Networking. Technical Report NDN-003, Name Data Networking, July 2012.
[9] M. P. Wittie, V. Pejovic, L. Deek, K. C. Almeroth, and B. Y. Zhao. Exploiting locality of interest in online social networks. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 25:1–25:12, New York, NY, USA, 2010. ACM.
[10] W. Wongyai and L. Charoenwatana. Examining the network traffic of facebook homepage retrieval: An end user perspective. In *Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on*, pages 77–81, 2012.