How to Make Content-Centric Networks Interwork with CDN Networks

Wei You, Bertrand Mathieu
Orange Labs
{wei.you, bertrand2.mathieu}@orange.com

Gwendal Simon
Telecom Bretagne
gwendal.simon@telecom-bretagne.eu

***Abstract*—The Internet architecture, based on end-to-end connections, had difficulties to efficiently deliver the always increasing number of contents. Content Delivery Networks (CDN) have been deployed to improve the delivery. Recent research works propose a new networking architecture, much more adapted to the current Internet usage (end-users just care about the contents they want and not about the endpoints that provide them). The Content-Centric Networking (CCN) aims at replacing the IP paradigm. CCN has been investigated by the research community for few years, and some demonstrators have proved its feasibility. However, less works have addressed the CDN use-case so far. In this paper, we show that the current CCN design, which does not allow negative reply, is not suitable for interconnection with the CDN service. We then propose the use of two new tables in the CCN nodes that are interconnected to the CDN servers, in order to detect possible misses in the CDN servers and subsequently forward the requests for missing contents toward the original servers instead of the CDN surrogates. The evaluations we have performed highlight that the integration of both new tables does not incur an increase of the memory requirement and thus a cost in the node. The proposed solution is thus a viable solution to make CCN network work in close cooperation with CDN networks, each one keeping its specific business models and functions: transport for CCN, storage for CDN.**

I. Introduction

The Internet success has changed the way we work, learn and play. Internet is widely used and leads to an increasing number of contents to be delivered by the network [1]. However, the Internet architecture, which has been designed in the 60s-70s and is based on end-to-end connections, faces a complexity to sustain the continuous traffic growth. Content Delivery Networks (CDN) were deployed in the last years to improve the content delivery, by replicating the most popular contents on some specific repositories, as close as possible to end-users. CDNs are now largely used and there is a clear business model for CDN providers and network providers: the CDN providers offer the storage function, while the network operators offer the transport function.

In the last years, network scientists have investigated a new networking approach, the Content-Centric Networking (CCN) [2]. This paradigm is based on the fact that nowadays users care more about the contents that they want but no longer about the endpoints who provide. Demonstrators have proved the feasibility of CCN, such as VoCCN [3], video delivery over CCN [4]. However, not many works address on the cooperation between CCN and CDN issue. Indeed, some researchers argue that CCN nodes includes native caching functions and thus can be CDN nodes themselves. But this way of thinking means that the hybrid CCN/CDN node should be able to ensure high capacity of routing and forwarding functions, as well as efficient content storage and management. As illustrated by [5] which concludes that a fast CCN router cannot support a big volume cache space or at prohibitive price, we believe that for performance issues, the two functions should not be mixed within one equipment. Furthermore, each actor (network operators and CDN providers) would not like to loose their current business models.

In this paper, we show the current CCN design is not suited to efficiently interwork with CDN repositories, because the native CCN design does not allow negative reply. In case of content miss in the CDN repository, the latter cannot inform the CCN node that it cannot serve this content. We then propose an evolution of the internal CCN node design, which is interconnected to the CDN repository. The evolution includes the use of two new tables: one to send requests toward the CDN repository (Repository Forwarding Table) and one to detect possible miss in the CDN server (Pending Repository Interest Table). Our evaluation highlights that the integration of both new tables does not imply more memory space than the current CCN node, and thus it does not incur a higher cost. We also study different implementing locations of our new node, namely *cRouter*, in a hierarchical topology. Since other researchers have different opinions regarding CCN/CDN interconnection, we present in a dedicated section, the others approaches and their advantages/drawbacks, compared to ours.

The rest of the paper is organized as follows. We detail the motivations, the cRouter architecture and the Interest/Data processing algorithms in Section II. We give some performance evaluations of our cRouter system in III. Then we discuss other solutions in comparison to ours in Section IV before concluding this paper in Section V.

II. cRouter: a extended CCN node supporting CDN repository

This section presents our cRouter solution, enabling the interconnection the CCN node with the CDN repository.

A. An overview of Content-Centric Network

The work about CCN is motivated by an observation that the Internet users now care more about *which* contents or information they are interested in than about *where* the information actually exist. However today's IP based architecture relies on a host-to-host conversation model. CCN aims at replacing this host-to-host design by a new client-to-content model.

In CCN, each object is independent from its location, identified by a content name, and every networking process, such as routing, content discovery and retrieving, is based on the content name. CCN utilizes two types of packet: *Interest* which contains a *ContentName* for sending the request of what a client wants and *Data* which contains also a *ContentName* for matching the Interest and replying the required content. The classic CCN node architecture contains three main elements:

- *FIB – Forwarding Information Base* is used to forward Interests to the sources that are known to potentially hold the matching Data. The FIB is filled by content advertisements which are brocadcasted by content providers. The structure of CCN FIB is similar as IP FIB.
- *PIT – Pending Interest Table* is to keep track of Interests so that the returned Data can follow the reverse path to consumers. When multiple Interests for the same content are received before the reply, with the purpose of saving the traffic, only the first one is forwarded. The others are pushed in PIT and wait for the Data back.
- *CS – Content Store* is a short cache or a buffer set in CCN nodes, to temporarily cache the Data objects.

The message process in a CCN node is as follow. When an Interest arrives at a CCN face (similar as interface concept in IP networking [2]), the ContentName is first checked in the CS. If the CS has the matching Data, the Data is directly returned. Otherwise the ContentName is checked in PIT. If there is a matching entry, the PIT is updated with the new arrival face. If not, the FIB table is consulted for forwarding. A new PIT entry with the ContentName and the incoming face of the Interest is created. When the CCN node receives the Data, the ContentName in the Data is checked in PIT. A PIT entry matching means this Data has been required, it is then sent out through the list of faces associated with the matching PIT entry, and the Data can be (optionally) stored in the CS.

B. Motivations for cRouter

In current CCN network, the servers must advertise about their contents. With such advertisements (*e.g.* the OSPFN [6]), other CCN nodes are aware of the available contents and where to forward the related Interests. Advertisements can work with a small number of contents but this solution cannot scale with current amount of different contents, which generates a big network overhead and a huge FIB. To overcome this drawback, since in reality CDN service focuses on a small set of content providers, all contents from the same domain names can be aggregated to a smaller identification name. Such idea has the potential to reduce the necessary memory size in CCN FIB and ease the forwarding process as well.

However, the CDN repositories hold only the popular contents offered by the service providers, not the full catalog. In CCN, if a node does not have the content, it either forwards the Interest out or ignores it, since there is no negative reply mechanism. Thus for CDN repositories, if we use aggregated information in the FIB, all the Interests related to this domain name will be sent towards the CDN repositories. But in case that the repositories do not have the contents (what we call *miss-hit*), the end-user never gets the content, whereas it is always available in the origin server. Thus our motivation comes from how to handle and to allow CCN nodes to cooperate with CDN repositories, and in case of a miss-hit, how to re-forward the Interests for this content directly to the origin servers and not again to the CDN repositories.

In our solution, a cRouter is *associated* with at least one CDN repository. A cRouter has the same functionalities as a regular CCN node and is connected to other CCN nodes. But it has two new components (Fig. 1) to connect with CDN repositories:

- *Repository Forwarding Table* (RFT), which stores the domain names of the content providers that have agreements with the associated repositories.
- *Pending Repository Interest Table* (PRIT), which stores the Interest messages that were previously forwarded to one of the associated repositories.

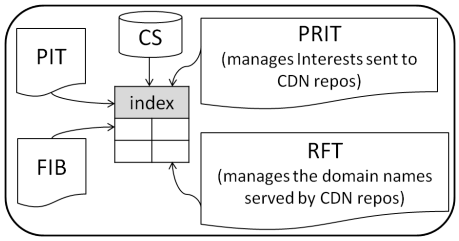

Fig. 1. The cRouter node architecture with the tables to manage repositories

C. The Repository Forwarding Table

Since we advocate the aggregation of contents in the advertisement phase for scalability issue and since the CDN caches do not have all the content (and it is dynamic), it is possible that we will get a miss-hit at the CDN nodes. But we cannot add the cache forwarding information into the FIB table neither, because in case of a miss-hit, we cannot differentiate the sources between the CDN repositories and the other potential providers. We propose to add in a cRouter a Repository Forwarding Table (RFT), which is used to support the CDN service. We define a RFT as a FIB-like table, which is used to store only the content domain names that are served by the repositories which are associated with the cRouter. As in a FIB, a RFT contains the outgoing face identifiers through which the CCN node can access the right repository.

To fill in the RFT, we see two options: (i) the ISP operator takes the responsibility of filling the RFTs of the cRouters . This option is for ISPs that know well their own CCN topologies and the repositories that are deployed in their networks, no matter these repositories are managed by themselves or by third-party providers. Although the set of contents that are actually stored in a repository can be dynamic, the domain names of these contents are pre-defined when the repository is installed, and are relatively stable. Furthermore, it is expected that the ISP well knows the configuration of cRouters and their associated repositories, hence RFT tables can be manually filled by ISP engineers themselves. (ii) the RFT is filled in the same way as FIB table does, *i.e.* through the implementation of a broadcasting protocols like OSPFN. But only the domain-name level name prefixes are published in the OLSA packets [6]. We propose to add a flag in OSPFN packets to differentiate the advertisements to FIB from those to RFT. When a cRouter gets a RFT-related advertisement, it creates

or updates a RFT entry with the domain names and the face identifier from which this cRouter received the advertisement. A regular CCN node receiving such RFT-related advertisement can ignore it. In the following, we detail RFT processing together with PRIT processing in Section II-E.

D. The Pending Repository Interest Table

In our vision, a repository is a storage server which does not implement any redirecting function (it is not a router). Thus, we have to implement a special mechanism to handle the miss-hit at the CDN repository, since CCN does not have negative reply. We propose the PRIT table, which is similar to the regular PIT table, but only applies to the Interest messages that have to be forwarded to repositories. The role of the PRIT is for sending back to users the content provided by the CDN repositories and also for detecting the incoming Interest messages that have probably generated a miss at a repository.

Each PRIT entry implements a timer (as in PIT) because CDN repository management can be dynamic and contents can be available later even if they were not for some time before. The role of the timer is then to clean the *content-miss* entries in the PRIT, where no Interest for this content has been received for a while. Contrarily to the PIT timer which is short (in the order of the Data round-trip-time), the PRIT timer should be set longer (for example in the order of minutes or hours), because CDN repository content can be dynamic but not at the scale of the second. Since the functionality of PRIT is similar as the PIT, in order to save the memory space, in reality implementation we can merge the PRIT and PIT together. For example we can add a third column which contains a flag for indicating PRIT information. This is only an implementation example but the principle does not change.

E. Packet Processing in the cRouter

We present now the Interest and Data packet processing algorithms, described with the different use cases. The Interest packet process with the different use cases is as follows:

Use case 1 *The Interest domain name is not present in the RFT (CDN does not cache content for this domain)*. The cRouter simply processes the Interest with regular PIT and FIB tables in order to forward the Interest to the origin server (as shown in Fig. 2).

Use case 2 *The requested content is cached in the CDN repository*. It is the first time the cRouter receives an Interest for the ContentNamen and there is no PRIT entry. The Interest is forwarded to the repository according to RFT. Meanwhile, a PRIT entry is created with the ContentName and the incoming face identifier (see Fig. 3).

Use case 3 *A matching entry in the PRIT but the incoming face is not within the associated face list*. The cRouter has already forwarded a same Interest to a repository and is still waiting for the Data. The PRIT only updates the entry with the new incoming face (as shown in Fig. 4).

Use case 4 *The incoming face is in the list of faces associated with a matching entry*. The cRouter receives again the same Interest from the client side, it finds a matching PRIT entry for both this ContentName and this incoming face. The cRouter thus realizes that the CDN repository did not reply for this content and a miss occurred. The Interest is then forwarded to the origin server according to the regular FIB. Moreover, the list of faces that was associated with the entry in the PRIT is now stored in the regular PIT, thus all previous Interests for the requested content will be satisfied when Data will get back from the origin server. The entry in the PRIT is kept in the table but *without any face* (we removed all associated faces). It thus indicated that a miss occurred and that future Interest should not be forwarded to repositories. Later if an Interest for this ContentName comes again, it will find a matching entry without any associated face and the Interest will be directly processed with the regular PIT and FIB processes (see Fig. 5).

Use case 5 *The CDN repository lately replies with a content*. This case means the CDN repository has the requested content but for any reason it replies late after the cRouter considered the case as a CDN miss. The first steps are the same as in use case 4, and when the delayed response from the CDN repository arrives at the cRouter, it finds a matched PRIT entry with an empty face list. In this case it simply deletes the related PRIT entry (see Fig. 6) to indicate it is not a miss and subsequent Interest messages can be forwarded to the CDN repository.

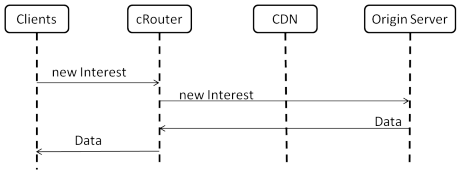

Fig. 2. Use case 1: The Interest domain name not present in RFT (this domain name is not managed by the CDN repositories.

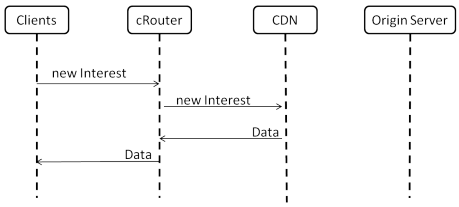

Fig. 3. Use case 2: Interest domain name present in RFT, but content name not in PRIT. CDN repository has this content.

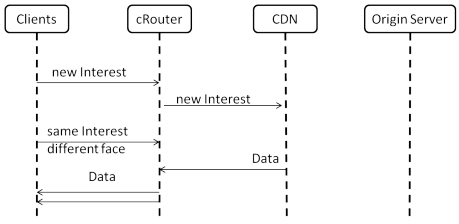

Fig. 4. Use case 3: Same Interest from different clients arrive, domain name present in RFT, CDN repository has this content.

The processing of Data packets at a cRouter is less complex. When a Data is returned back to a cRouter, we first distinguish whether this Data comes from the repository or from a origin server according to the incoming face. If the Data comes from a origin server, the regular PIT lookup process is performed. The related PRIT entry (if it has any) is not deleted

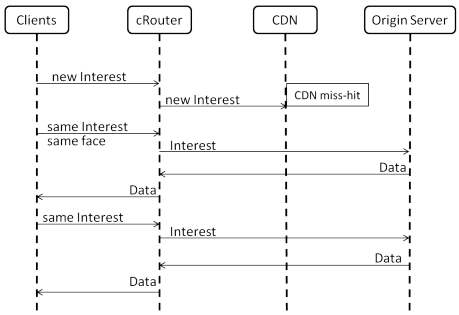

Fig. 5. Use case 4: CDN repository miss (domain name managed by the CDN repository but content not cached).

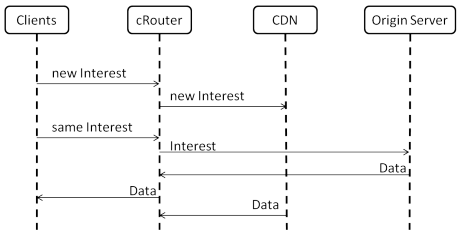

Fig. 6. Use case 5: CDN repository replies lately

so that the following Interest messages will not be sent to the repository again. If the Data comes from a repository, only a PRIT lookup is performed. Thus, the Data is sent to all faces that are associated with the ContentName in the PRIT, and then the entry is deleted.

III. Evaluations

We now evaluate the cost of implementing cRouters. We suppose a cRouter which is connected to the CDN repositories with a line cards of 10Gbps capacity. The Data *round-trip-time (RTT)* for in-domain networking and for out-domain are 20ms and 150ms respectively [7]. We consider that 65% of the incoming traffics can be forwarded to a repository (the domain names are handled by the CDN) [8]. We assume each Interest packet has 40-Byte length, the length of CCN ContentName is 30-Bytes and we add 2-Bytes for the face identifiers in PRIT entry [5]. We consider today's memory chip technologies (Table I) for a summary from [5]) and two possible implementations of the PRIT. The most basic implementation is based on hash table, while a more sophisticated implementation is based on Bloom filters [9]. We applied a counting Bloom filter which has 5 independent hash functions and the designed false positive rate is 0.1%.

A. Memory requirement analysis

A cRouter contains two new tables. We first measure the extra-cost of the memory chip. Since more information are managed, it takes more memory space. We focus on the PRIT in this discussion, because the RFT stores only a few domain names. According to the studies [8], a large portion of the traffic volume is related to a dozen of domain names, so the size of RFT is only in the order of hundreds Bytes, which can be implemented on any fast memory chip.

The additional PRIT table introduce extra memory cost but the PIT size is also reduced since the Interests that are

Technology	*Access time (ns)*	*Cost ($/MB)*	*Max. size*
SRAM	0.45	27	≈ 210 MB
RLDRAM	15	0.27	≈ 2 GB
SSD	1,000	0.03	≈ 10 TB

TABLE I. Current memory technology [5]

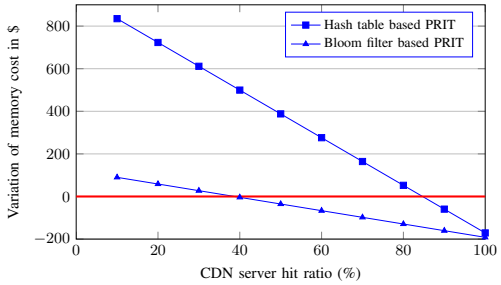

Fig. 7. The memory cost variation vs. the cache hit ratio

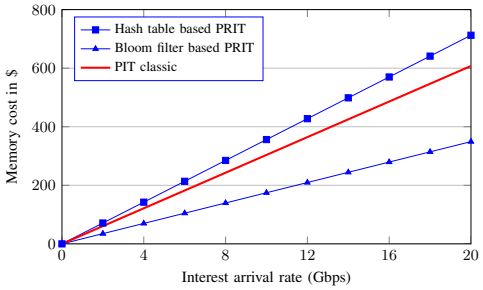

Fig. 8. The memory cost vs. the Interest arrival rate

appended in PRIT no longer exist in PIT. We represent in Fig. 7 the variation of total memory cost in function of the CDN hit ratio, compared with a single CCN PIT cost (the red solid curve in Fig. 7). When the CDN hit ratio is only 20%, the hash table based PRIT costs $723 more, the Bloom filter based PRIT costs less but still $58 more than a regular PIT table. But when the CDN repository can offer 80% hit ratio, the additional cost for hash table based PRIT is only $50, and the Bloom filter based PRIT can even save $130. The hit ratio plays an important role. Indeed, the in-domain latency between CDN repositories and cRouters is smaller than the out-domain latency between ISP peering routers and origin content servers. Thus according to *Little's Law*, the more requests are treated by repositories, the less Interests stay in the PRIT, so the smaller is the table size.

In Fig. 8, we show the total memory cost in function of the Interest arrival rate. Here we vary the $\lambda_{interest}$ from 1Gbps to 20Gbps. The CDN hit ratio is fixed as 80%. We can observe that the Bloom filter based PRIT always cost less memory than the classic CCN PIT. And the more the traffic is intensive, the more saving we get. The hash table based PRIT cost more memory when the $\lambda_{interest}$ increases.

B. Location of deployment analysis

We consider now a hierarchical topology with 3 levels: the peering level, the regional core level and the edge level. For an ISP, CDN repositories can be located at either the regional level or the edge level. Since regional nodes experience a higher

packet rate, cRouter should implement fast but expensive memory chips (*e.g.* SRAM in Table I)) for PRIT, but less CDN repositories are required. On the contrary, RLDRAM can be enough for PRIT at edge nodes level but more CDN repositories are needed.

We now set the Interest arrival rate of *edge node* in the range of $1 - 10$ Gbps. A repository is equipped with a 1TB high-speed SSD disk. We evaluate the cost of the PRIT memory with such traffic. See Fig. 9. Until a 7Gbps Interest arrival rate, the regional cRouter deployment is cheaper because, in both locations, cRouters are equipped with RLDRAM-based PRIT but the overall memory size of cRouters at the regional level is smaller than at the edge level. However, when the Interest rate exceeds 7Gbps, the regional cRouter deployment becomes more costly because the regional node collects all the traffic from the edge nodes and the traffic rate becomes so high that SRAM is the only choice for PRIT.

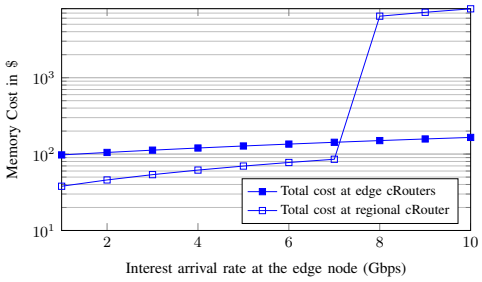

Fig. 9. The cost of different deployment location

IV. Discussions

With regard to the literature related to CCN, we understand that many researchers believe that the CCN on-path caching will eventually replace CDNs. We think there is a misunderstanding about today's Internet traffic reality. As we explained in Section I, it is not a good idea to make the routers process a high volume of traffic and manage a huge content storage. In our opinion, the main role of CS in CCN nodes is to improve Data packets retransmission in case of packet loss. Such role differs from the one of a repository, which leverages complex strategies to optimally utilize a high volume storage.

In the same way, an idea is to transform the CDN repository into a transit CCN node with an extra-large Content Store (*e.g.* the *CCNx Repository [10]* element). Thus, the CDN/CCN node simply forwards the Interest as in a regular CCN nodes (based on its PIT and FIB tables) in case of *miss-hit*. We discarded this idea because: 1) the networking topology is impacted; 2) a failure of the CCN/CDN node results in a loss of all Interests, since the node is on the forwarding path; 3) the repository should perform not only a fast content management, analysis and retrieving process, but also a fast Interest management and redirection performance, which is not viable [5].

Another option is to leverage native multicast features of CCN. Indeed, the FIB can have several outgoing faces for an entry and thus multicast the Interest message to all faces. We can then have one face for the CDN repository and one for the origin server specified for the domain name, served by the CDN network. The Interest can be simply forwarded twice. However, this solution does not satisfy neither network operator nor the origin server, which receives all requests although most of them are treated by repositories. Such solution doubles Interest-related traffic. To detect CDN miss, it is also possible to ask the CDN to resend the missed Interest back to the CCN node. Once CCN node receives an Interest from the face that is reserved for CDN repository, it knows that a CDN miss occurred. This method relies on the CDN repository status, which means that if the CDN node is down, all the Interests for the related domains are absorbed by the "*hole*" and the CCN node cannot be aware. We prefer to rely on clients to detect the miss events.

V. Conclusion

CDN are used by many content providers and have proved the efficiency for the current Internet. Meanwhile the CCN paradigm emerged, with some use-cases to show the feasibility of the approach, but did not take into consideration CDN networks so far. In this paper, we showed that the current CCN design can not support and interconnect with CDN service and an evolution of the CCN node interconnected to the CDN repository, is necessary. That is what we have proposed our cRouter, a CCN node with two new tables, namely the RFT (Repository Forwarding Table) to route the request toward the CDN repository and the PRIT (the Pending Repository Interest Table) to detect possible miss of contents. The evaluations we have performed show that including the two new tables has no additional cost for implementation. We have also exemplified how our cRouter can be deployed in a hierarchical network of an operator. Our cRouter is fully compliant with current CDN nodes and allows to keep the relationships and positions of key players in the delivery chain: The CDN providers still keep the control of the storage functions, while the network operators still focus on the transport part. In the future work, we plan to develop the software of the tables in the CCNx software and to validate this solution at large-scale.

References

[1] Ericsson mobility report: on the pulse of the networked society. Ericsson, Nov. 2012. http://www.ericsson.com/news/1659597.

[2] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N Briggs, and R Braynard. Networking named content. ACM CoNEXT, 2009.

[3] 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 *ACM Re-ARCH workshop*, 2009.

[4] Y. Liu, J. Geurts, Point J-C., B. Rainer, S. Lederer, C. Müller, and C. Timmerer. Dynamic adaptive streaming over ccn: A caching and overhead analysis. In *ICC*, 2012.

[5] D. Perino and M. Varvello. A reality check for content centric networking. In *ACM Sigcomm workshop on ICN*, 2011.

[6] L. Wang, M. A. Hoque, Ch. Yi, A. Alyyan, and B.C. Zhang. OSPFN: An OSPF Based Routing Protocol for Named Data Networking. Technical Report NDN-003, Name Data Networking, July 2012.

[7] D. Saucez, A. Kalla, C. Barakat, and T. Turletti. Minimizing bandwidth on peering links with deflection in named data networking. March 2012. INRIA paper.

[8] V. Gehlen, A. Finamore, M. Mellia, and M. Munaf. Uncovering the big players of the web. In *Traffic Monitoring and Analysis*. 2012.

[9] W. You, B. Mathieu, P. Truong, J. Peltier, and G. Simon. Dipit: A distributed bloom-filter based pit table for ccn nodes. In *ICCCN*, 2012.

[10] CCNx Repository. http://www.ccnx.org/releases/latest/doc/technical/RepoProtocol.html.