Exploiting end-users caching capacities to improve Content-Centric Networking delivery

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

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

***Abstract*—A Peer-to-Peer (P2P) network is decentralised and relies on end-users capacities for a good delivery of service. The more end-users are sharing the content, the more powerful the system is. P2P applications are now widely used for many services. But P2P systems are still based on a end-to-end connectivity on the current IP-based Internet. For few years, a new networking paradigm emerged, the Content-Centric Networking (CCN) approach. In CCN, the user just cares about the content, not about the node that provides it. It takes as input some features of the P2P systems, applying them in the network itself. However, current CCN system forwards messages based on a defined forwarding table, filled with the advertisements of content providers. It does not take into the consideration of the fact that end-users that requested the contents can cache them for a while and thus be the potential sources of content, as it is in a P2P system. In this paper, we propose to take into consideration of the caching capacities of end-users in the CCN design so as to improve the content delivery in a CCN network. This is achieved by the design of a content-aware dynamic downstream forwarding scheme, and an adaptation of the current way to fill the forwarding table. Our evaluation shows that exploiting the end-users P2P caching facilities allows to improve the efficiency to the CCN network by reducing the memory requirements of the CCN node and by improving the data delivery to end-users via a closer delivery.**

I. Introduction

Peer-to-Peer (P2P) systems have been widely used for more than a decade. For instance, millions of users share files through Bittorrent system [10, 16]. P2P systems have also been implemented for internal purposes in data-centers, typically Bittorrent protocol is used by applications such as Facebook [3] or Twitter [1]. Another success story of P2P applications is Spotify [8, 13], which attracts millions of users for a music streaming system. In Asia, live video P2P applications such as PPLive, PPStream, and Sopcast are also popular. Those P2P applications get full benefits of their decentralized nature, which enables a significant reduction of server load and bandwidth costs. They also benefits from end-users capabilities, such as storage and bandwidth. P2P grew very quickly because the network is currently just "a set of dumb pipes", interconnecting endpoints together, via the IP protocol, without any added-value in the network.

Today's Internet users tend to be more interested in the content that they want to download than in the location of this content. This mismatch between this usage and the original design of Internet calls for a radical communication model change, which is centered on information access. An emblematic representative of such radical change is Content-Centric Networking (CCN) [12], which is a novel network architecture revolved around the production, consumption and transformation of information matching user interest. CCN is a shift from a host-based networking to a content-based networking.

CCN is a receiver-driven model, where users' requests are expressed by an *Interest* for a given object. Interests are routed in the network based on their names toward a node having or processing it, and where the related *Data* is sent back along the reverse path, with the possibility for intermediate nodes to cache the object. The forwarding information, which is based on names, are cached in the Forwarding Information Base (FIB) table of each CCN node. This table contains the prefixes of the ContentName and the outgoing interfaces (*a.k.a.* faces in CCN technology) information. The reception of advertisement messages allows each CCN node to fill its FIB. To decide Interest routing, CCN nodes look for the longest name prefix match in their FIB.

Although CCN is considered as a promising system, it still has two main limitations. First, the FIB can be huge in case of many different contents, advertised by many content providers, because no context-awareness (*e.g.* content popularity) is performed with current advertisement protocols. Second, CCN does not lever the caching capacity of end-users, as it is in P2P applications. When looking for a content, CCN node resolve content name in their internal forwarding table, namely FIB. However FIB are filled in by content advertisements, which are generated by contents providers. The CCN forwarding scheme ignores that other end-users may have previously requested similar content, thus they can be source for the content.

In this paper we propose a CCN content-aware dynamic forwarding design, which exploits P2P end-users caching capacities and simplifies the advertisement process, with:

- *A dynamic Interest downstream forwarding scheme*. Our goal is to make a better use of CCN Data packets that are cached. To this end, we propose a new dynamic Interest downstream forwarding scheme, which aims at looking for cached contents from downstream nodes, including end-users' devices.
- *A content advertisement protocol*. Advertisement are not propagated into the entire network, but only via certain

paths with respect to the network topology. Thus different CCN nodes do not have the same view of the entire available contents.

- *A new algorithm to fill the FIB*. In our proposal, FIB is not filled from broadcast advertisement messages but rather from incoming Data packets in a dynamic manner. Hence FIB memorizes only the content name prefixes or the domain names that are locally popular.

The rest of the paper is organized as follows. We introduce the CCN paradigm in Section II before presenting our content-aware dynamic forwarding system in Section III. We evaluate the feasibility and performance of our proposal in Section IV. Finally, we present the related work in Section V before concluding.

II. Content Centric Networking

The works about CCN (as well as other proposals related to information-centric approaches) are motivated by the observation that the Internet users now care more about *which* content or information they are interested in than about *where* the information are. However today's IP Network architecture relies on a host-to-host conversation model. The CCN proposal is an attempt to replace today's host-to-host design by a new client-to-content model.

A. Named Data Networking

In CCN, each networking element is no longer identified by its location, but by a content name. CCN uses a hierarchical, structural, human-readable, unbounded name (*e.g. ccnx:/orange.fr/news/video.swf/v1/s0/*). Each networking process (routing, forwarding, content discovery and retrieving) is also based on content name. CCN contains two types of packet: *Interest* for sending the requests and *Data* for replying the matching content. Each Interest packet carries a *ContentName*, which expresses what the client wants. Each Data packet carries also a *ContentName*, which is used to describe the content in this Data and to match the Interest.

B. Architecture of a CCN node

Figure 1 illustrates the architecture of a CCN node, and also the processing when Interest and Data come. One CCN node is composed by three elements:

- *FIB – Forwarding Information Base* is used for forwarding the Interests to the sources that are known to potentially hold the matching Data. The FIB is filled by content advertisements messages brocadcasted by content providers in the CCN network. The FIB is almost identical as the IP forwarding table.
- *PIT – Pending Interest Table* is for keeping the tracks of propagated Interest so that the returned Data can follow these tracks downstream to the consumers. The second role of the PIT is to prevent that multiple incoming Interest packets generate multiple packet forwarding. When Interests for the same content are received, only the first one is forwarded, the others will be only pushed in PIT and waiting for the Data back.

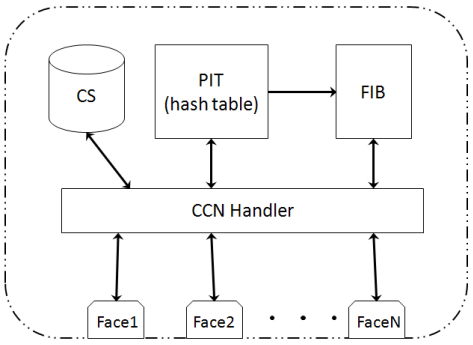


Fig. 1. Original CCN node

- *CS – Content Store* is a cache or a buffer set in CCN nodes. The in-network caching system of CCN leverages on the next-generation routers, which incorporate some caching memory for the goal of minimizing network bandwidth and latency, as well as the server occupancy.

C. Message Processing in CCN

When an Interest packet arrives at a CCN face, the ContentName carried by this Interest is firstly checked in the Content Store. If there is a matching Data, it will be directly returned through the same face where the Interest arrived at. Otherwise the ContentName is further checked in PIT. If there is already a matching entry, the arrival face information is updated in the matching entry. If not, the FIB table is consulted for the forwarding information, and a new entry associated with the new arrival Interest and its incoming face is create in PIT.

The Data packet follows the "*bread crumb*" left by the Interest in the CCN nodes. When a Data packet arrives at a CCN node, the ContentName of the Data is checked in the Content Store. If there is already a matching content cached in the CS, this Data packet should be discarded. If there is no existing match in CS, the Data is checked in PIT. A PIT entry match means this Data has been required, it is sent out through the list of faces associated with the matching PIT entry, and the Data can be (optionally) stored in the Content Store.

III. A Content-Aware CCN forwarding structure

As previously seen, current CCN FIB is filled by advertisements messages broadcasted in the CCN network from the providers that propose contents. But the messages are advertised network-wide without context-aware information (where the content is really requested, popularity of the contents at various location, *etc.*). Furthermore the FIB does not take into account the previous *Interest* requests passing via the node.

We present now our dynamic content-aware forwarding solution for CCN. Our proposal requires a new architecture for the "forwarding unit" of CCN node. The new structure is represented in Figure 2. We add a new table, which is called Dynamic Interest Forwarding Table (DIFT). We also significantly change the way FIB is filled. To avoid confusion between today's FIB implementation and our proposal, we say hereafter "content-aware FIB". The fundamental idea behind

our proposal is that we do not use broadcasted advertisement messages to decide the forwarding. Instead, we leverage Data delivery paths, which allow each CCN node to take into account the local popularity and availability of data.

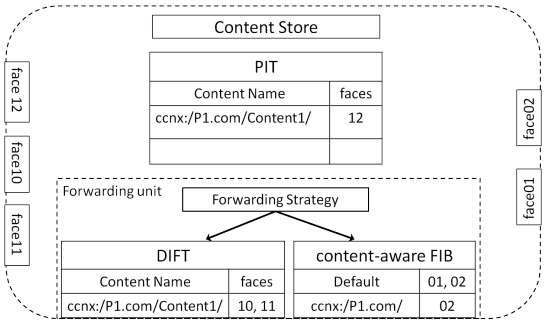


Fig. 2. The node structure of the content-aware forwarding design

A. Content Advertisement Protocol

We aim to realize a content-aware forwarding solution but it does not mean that we do not need any content advertisement mechanism at all. Content providers still need to advertise content, otherwise the network will not be aware of what providers offer. However, in our proposal, these advertisements are not broadcasted in the entire network. Furthermore, CCN node does not rely on these advertisements to fill content-aware FIB. In the following, we use indifferently content advertisement and content publication, which is a common term in network.

The *default face* is an important concept in our content advertisement protocol. The default face(s) at each CCN node are the interfaces that are connected to the "upper" node, which is the next routers on the path to the peering nodes of an Autonomous System. It can typically be the root nodes of a hierarchical networking topology or the border routers of a mesh networking topology.

The configuration of the default faces can be realized either manually, or through the Shortest Path algorithm. For example in a hierarchical networking topology, the network operator knows how nodes are connected with the upper node(s), the neighbour nodes. Thus Internet Service Provider (ISP) can easily configure the default face(s). In a mesh topology, the default face(s) can also be configured so that the shortest path to the border nodes are default faces.

Let say a content provider wants to advertise a new content *C*. The equipment that is in charge of emitting content publication is the first router of the content provider delivery infrastructure. The first router first adds the publication information into its FIB. Then it propagates the publication through all its *default faces*. The next nodes receive this publication, and perform the same process, *i.e.*, add the publication into their FIBs and forward it through the default faces. Finally the publication reaches the border nodes of the Autonomous System of content provider. Thus the nodes that are on the publication forwarding path and the border nodes have the published content *C* in their FIB. The border nodes also collect all content publications of other connected ASes.

Network operators can decide the number of border nodes in their AS, according to network topology and size. Overall, our proposal matches a fundamental principle of Internet, which is to let each network operator manage its own network. Here, the diffusion of content advertisements actually depends on the policy of each network operator. Other ways to collect and exchange content include distributed implementation based on Distributed Hash Tables (DHTs).

B. Content-Aware Dynamic FIB

To fill a content-aware FIB, many options are possible. Our proposal is to leverage incoming Data messages. When a CCN node receives a Data message, it memorizes the domain name of the ContentName and the incoming face in content-aware FIB. It can thus forward the subsequent Interests for the same domain name through the face from which it received the Data to reach the content container.

The algorithm of the incoming Data at the content-aware forwarding structure is shown in Figure 3. The content-aware FIB does initially not contain any forwarding information, except the default face(s) configuration. Once the node receives a content packet, it retrieves the domain name and adds it into FIB together with the incoming face information. If the content-aware FIB does not have any entry on this domain name, it creates one. If there is already an existing entry on this domain name, it updates the entry with the new arrival face. It is allowed to have several outgoing faces. Typically contents from a given domain can be provided by different providers, for example the origin server or the decentralized P2P storage nodes.

Each entry is thus associated with a face list, which is updated at reception of every matching Data packet. We opt for a simple First-In-First-Out (FIFO) replacing strategy to avoid too huge face lists. If one entry has too many faces in the list, the older faces should be replaced with the newer ones, so that memory space is saved and expired information are discarded, For the same reason, the entire content-aware FIB also includes a replacement strategy (LRU or Random) for the entries (different domain names).

When the node receives an Interest request, if this Interest matches the content-aware FIB on the Interest domain name, this Interest is forwarded through the faces of the matching entry. If the domain name is unknown by FIB, the node sends them through the default face(s).

C. Integration of CCN in the P2P application

Current P2P applications work with IP. To be usable with CCN, they should integrate the CCN protocol stack and mainly the CS function the node should manage. The internal CS within the P2P application can be the local hard-disk (or a huge memory) where to cache contents. Since the P2P application is deployed in the end-user devices, with less traffic passing through them, compared to a network router, contents will stay longer in the CS. Thus it is a great opportunity for a CCN

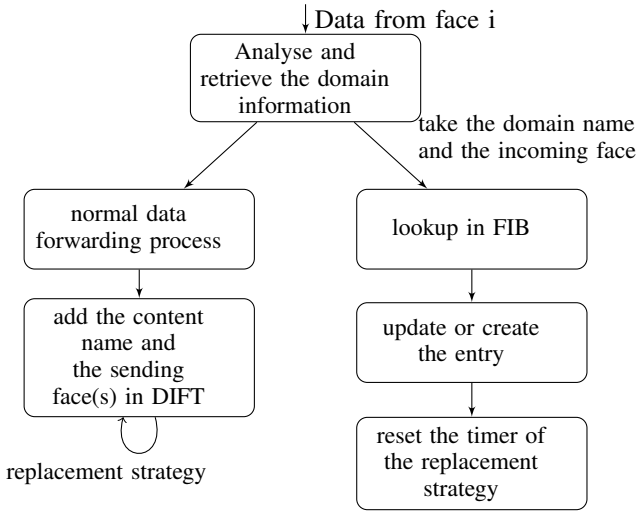


Fig. 3. Algorithm for the processing of incoming Data packet

network to be able to use those P2P end-users CS, meaning an access to caches content with a high possible hit-ratio. However, in the current CCN model, the Content Store does not announce in the network which contents it holds, thus the P2P application being at the end of the delivery chain, it can not serve as a potential content source for close requesters. In our proposal, the integration of the CCN functions in the P2P application, together with the DIFT mechanism, which is explained in the next section, allows the CCN network to access contents cached in the P2P CS.

D. Dynamic Interest Forwarding Table

In addition to the above algorithm for content-aware FIB, we propose another component and protocol to make a better use of Content Store for P2P. In this paper, we use the generic CCN term Content Store (CS) to mention both the routers buffer or the P2P end-users caching capacities.

We propose a new table, which is called Dynamic Interest Forwarding Table (DIFT), to take advantage of the downstream CSs. An DIFT stores recent Data packets that have been forwarded to downstream nodes. The idea is that, since downstream nodes have generally to manage less traffic, it is possible that a request for a content that has been recently treated by a node is still available in the CS of one downstream node. In other words, contents cached in CS are expected to stay longer in downstream routers, so routing Interest messages to them makes sense. It is even more true for content cached in the end-users caching capacities, as we have with P2P applications.

The structure of DIFT is similar as the PIT. Each DIFT entry contains two columns, one is for the ContentNames and the other one is for the related face identifier(s). When a Data packet is returned to a CCN node, a PIT lookup process is performed. The Data message is forwarded through the faces of the matching PIT entry, and the matching PIT entry is then deleted. DIFT table is used to temporarily memorize the deleted PIT information. It may serve in a very short term to forwarding Interest (as shown in Figure 3).

We give in Figure 4 the flowchart for Interest processing in our content-aware forwarding algorithm. When an Interest arrives at a CCN node (and requested content is not stored in the local CS), after the PIT process, there are two forwarding steps: DIFT lookup and content-aware FIB lookup. If a matching entry is found in DIFT, it means that one copy of the requested content has been recently sent to some downstream nodes, so it might still be stored in their CS. Hence, the Interest packet is forwarded to those downstream nodes. FIB.

How to use the DIFT table and the content-aware FIB together can vary and depend on a forwarding strategy based on the concrete networking conditions and eventually network operators policies. Typically, when a node receives an Interest, it can choose whether to only use the DIFT, to use the DIFT or not, or to use both, in parallel or serially. The motivation for sending the *Interest* towards upstream nodes (as current CCN) is that the content might not still be in the CS of downstream nodes and end-users caching capacitites, and thus the CCN node forwards the *Interest* towards a reliable content provider to ensure that the content is delivered to the end-users. That is the role of the strategy bloc we add between the FIB and DIFT in Figure 4 to define at it works.

The DIFT is also implemented with a purge timer, because the behavior of DIFT is far more dynamic than content-aware FIB. First, it memorizes longer name prefixes while FIB stores only domain names. Second, the forwarding information in the DIFT is related to the dynamic content container (*e.g.* the CS of the downstream CCN nodes or the end-users of a P2P system), while the content-aware FIB is related to the relatively reliable content providers (*e.g.* the content provider, the CDN repositories or the ISP caches). Since the content sources that are included in DIFT are not as reliable as those in content-aware FIB, the entries at the DIFT should be updated and refreshed faster with a simple replacement strategies (for example the FIFO or Random policies).

IV. Evaluations

We performed some evaluations of our proposal to study the feasibility of our content-aware CCN forwarding scheme, with a focus on memory in CCN nodes and on the response time. The motivation is that previous works on CCN nodes have revealed that memory issues within CCN node can seriously prevent the implementation of some smart but not realistic algorithms, *e.g.* [15, 20]. We thus decided to have a comprehensive look at memory implementation issues.

A. Settings

We consider a typical CCN node, which is deployed in a network where in average each node has 3 external links (*e.g.* the Abilene topology). Moreover we consider one *default face*. We suppose there are 12,000 content that are advertised in the network, including 2,000 which have been received by this peculiar CCN node. The node receives in average 200

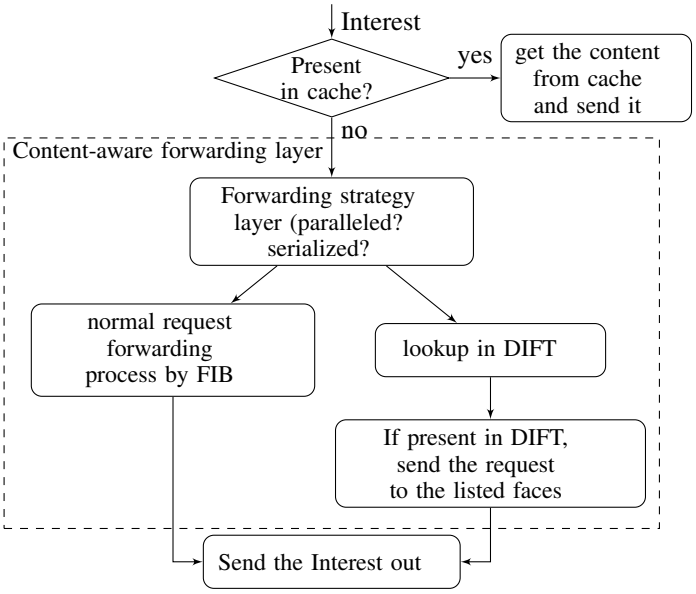


Fig. 4. Algorithm for the processinf og incoming Interest packet

Interest per second and each Interest experiences a 10 ms Data RTT. The size of each entry size in the traditional FIB and in the DIFT are 32 Bytes (30 Bytes for the ContentNames plus 2 Bytes of the face identifiers), and the average entry size of content-aware FIB is 22 Bytes (in other words the domain name is in average 22 Bytes long).

The popularity of the set of contents is a critical parameter. We assume a Zipf popularity distribution with a parameter α. Previous works have revealed how critical is this parameter [17]. In the following, we make this parameter vary in typical ranges from 0 to 2.

B. Impact of Content Popularity

In Figure 5, we present the memory requirement according to the parameter α of the Zipf distribution. In the traditional CCN publication algorithm, the CCN node receives all content advertisement, and FIB is filled accordingly. Hence the size of classic FIB is independent of α. Our content-aware FIB is of course smaller. We evaluate this in Figure 5.

We first analyze the solution with only content-aware FIB. The size of this content-aware FIB is the highest for small values of α. The more uniform is the content popularity, the higher is the probability that the CCN node has to forward a Data related to every content in the catalog, and consequently the larger is FIB size. When α increases, some contents are never requested, so this CCN node ignores these contents.

We now study the association of content-aware FIB and DIFT. We assume every DIFT entry stays 10,000 time Data RTT. As can be expected, the size of DIFT is more sensitive to content popularity. The overall memory requirement is close to the one required by the traditional FIB implementation when α is around 0.8, which corresponds to the generic web traffic [7]. For values of α bigger than 0.8 (*e.g.* it is case for VoD traffic [7]), the overall memory requirement is smaller than the traditional FIB.

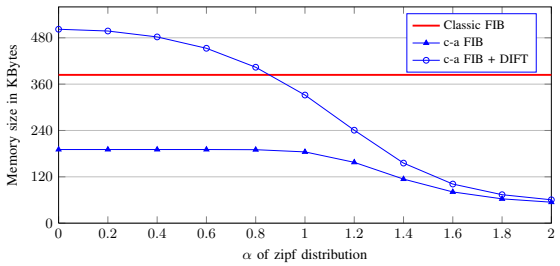


Fig. 5. Memory requirement vs. content popularity

C. Impact of Network Topology

We now explore the impact of network topology on memory requirements. In Figure 6, we represent FIB size according to the degree of CCN node. Different curves represent different values of α for content popularities. It has to be recalled that Data packets coming from the default face are not memorized. A node with many connections to other CCN nodes should thus store more Data packets because the overall traffic coming from non-default face is bigger. In current core networks, each core node is in average connected to 3-4 links (*e.g.* Abilene topology or Géant topology). In this case, our content-aware FIB consumes not more than 50% of the classic FIB memory.

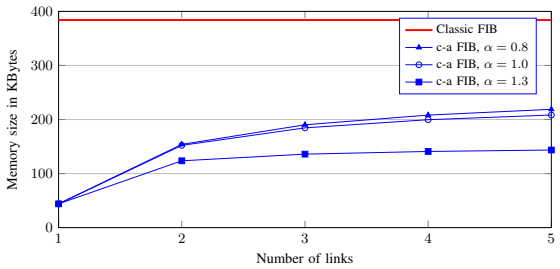


Fig. 6. FIB memory requirement vs. the number of links

D. Impact on Entry Storage Duration in FIB

FIB incorporates a timer. Entries are automatically removed from FIB when they are too old. In Figure 7 we analyze FIB size according to the entry saving duration. For small values of α, the table size reaches it maximum even with a shorter saving duration, that means the FIB is filled fast. On the contrary, for example when $\alpha = 1.3$, even if entries are kept for 30 minutes, FIB does not reach its maximum value. This is can be explained by observing that FIB is filled much slower when content popularity is less uniform.

E. Impact of Catalog Size

In our previous evaluations, the size of catalog (*i.e.* the number of different contents) is set to 12,000. In Figure 8 we analyse the impact of the number of publications on the memory size requirement. Intuitively, more publications means more memory space needs. We can see the size of classic

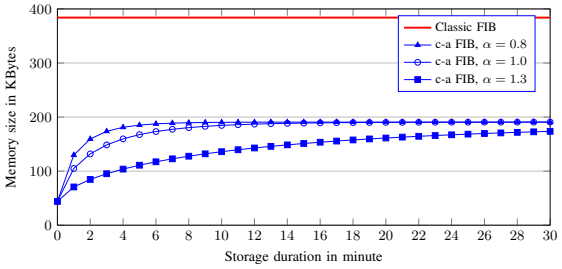
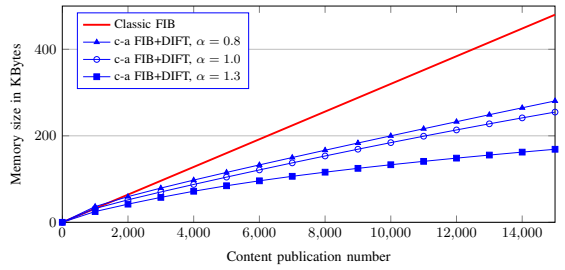
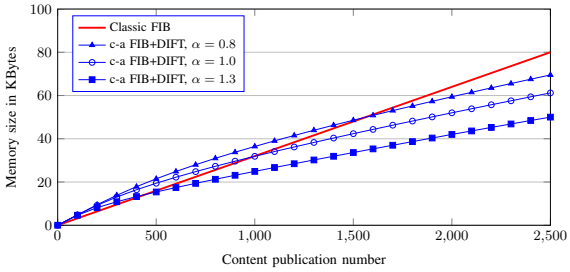


Fig. 7. FIB memory requirement vs. the FIB entry storage duration

FIB linearly increases with the number of publications. In Figure 8(a), we show a wide evaluation with potentially very large catalog. Here, our content-aware FIB enables a significant gain in terms of memory requirements. In Figure 8(b), we emphasize the special case of very few different contents. Here, the traditional FIB can be more efficient than our content-aware FIB only for $\alpha = 0.8$. Otherwise, our solution is always at least close to the traditional FIB, or far better.



(a) zoom out



(b) zoom in

Fig. 8. Total memory requirement vs. the number of publications

F. Impact of DIFT hit-ratio

The Content Stores of downstream nodes have their own replacing strategies. Of course, all the Interests that are sent by the upstream DIFT table do not generate a match. In this evaluation we suppose the downstream nodes can offer a 75% CS hit-ratio. Thus we define the DIFT hit-ratio as the number of incoming Interests that gets a DIFT match and get the downstream CS hit, over the number of all incoming Interests. In Figure 9 we see that applying a bigger DIFT size can offer a better DIFT hit ratio. And a bigger Zipf α can raise also a better DIFT hit-ratio. For example the Zipf distribution of the generic web traffic has an $\alpha = 0.8$, the 64 KBytes, 128 KBytes and 256 Kbytes DIFT can offer a 49%, 58% and 68% DIFT hit-ratio, respectively.

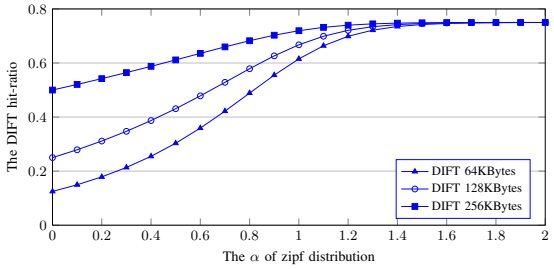


Fig. 9. DIFT hit ratio vs. the content popularity distributions

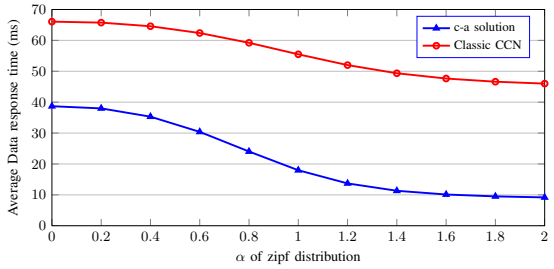


Fig. 10. Average Data response time vs. the content popularity distributions

G. Impact of response time

With our strategy, some requests are forwarded downstream to P2P users according to DIFT table. If P2P users are closer, the content delivery time can be improved. We analyze hereafter the average Data response time with regard to the content popularity.

We suppose a hierarchical networking topology, which has three levels: the root nodes, the core nodes and the edge nodes. The average round-trip times are 10 ms for links between levels, and 80 ms to reach the origin servers. For example if the content-aware DIFT has a matching content at P2P end users that are in the same sub-network as the requester, the Data delivery time is 10 ms (respectively 20 and 30 ms) if the root of this sub-network is an edge node (respectively a core node and a root node). The Content Store of each node can offer 0.3%, 0.2% and 0.1% content hit-ratio at edge node, core node and root node, respectively, since down level nodes receive less intensive traffic than high nodes. If the content request gets satisfied at the Content Store of the node of a certain level, the delay is thus half of the related Data round-trip time.

In Figure 10 we can see that Interest experience a shorter response time when our content-aware forwarding solution is implemented. This result is achieved for any content popularity We also observe that the difference between content-aware solution and the traditional CCN forwarding is greater when α is large than when it is small. This can be explained as the more popular a content is, the larger are chances that Interests are served at the downstream nodes, which offer a shorter response time.

V. Related Works

So far, previous works related to CCN have addressed improvement of Content Store performances [4, 14], techniques to make CCN nodes more feasible [6, 15, 20] and FIB forwarding processes [11].

But no previous work propose a downstream Interest forwarding structure as we proposed in this paper. In the CCNx implementation, a repository node[2], a CCN node with a disk storage capacities is proposed, to offer a larger caching space (compared to traditional Content Stores of routers) but no improvement to the CCN node structure to optimize the use of the repositories is proposed. Our solution, being improved for the use of end-users caching capacities is then innovative and might be used jointly with the repository, the latter being seen as the P2P end-users caches.

How to advertise content and how to fill the FIB are still open issues. Current information-centric forwarding solutions are based on either adding broadcast type messages, or including an additional element in the network [5]. These methods try to passively forward requests to the potential content locations without actually considering the name-based characteristics of data. To broadcast content advertisement, the NDN project [21] proposes OSPFN [18], which is an extended version of the IP OSPF protocol for CCN. OSPFN uses the Opaque Link State Advertisement (OLSA) to announce content names or prefixes and router IDs. A CCN node can use the OSPF protocol to build its local Link State DataBase (LSBD) and then inject information into the FIB. OSPFN is the first relatively mature proposal related to content advertisement in CCN. But we are afraid this proposal is not CCN-friendly for three reasons. First, each OLSA is propagated through the entire network and reach every CCN node. Such implementation generates a lot of OLSA traffic into the whole network. Second, each node has the same and entire view of the whole networking topology. This design matches needs of IP networks but it is inappropriate for CCN, since CCN nodes look for contents only when they receive requests for them. Indeed some contents are only popular at certain geography regions [19], and different regions do not have the same catalogue of popular contents [9]. This calls for CCN nodes having a local view of contents availability. Third OSPFN is still a location-based broadcast algorithm, which relies on router ID. These three points motivates the exploration of new content-aware forwarding solutions—the content-aware dynamic CCN forwarding.

VI. Conclusion

We present in this paper a content-aware dynamic forwarding proposal for CCN, including the consideration of end-users caching capacitirs for improving the CCN approach. In comparison to the traditional broadcast-based approach, our approach is more respectful of the information-centric principles. Our content-aware forwarding proposal let each CCN node only store domain names that are close to it, thus FIB information differs among CCN nodes to reflect localized variation of popularity. In our approach, we also make a better use of the caching capacities of the downstream nodes, being the caches of the end-users, as with P2P applications or the content stores of CCN routers. Our evaluation demonstrates that such CCN node implementation is feasible with regard to memory requirements, with table size that is not larger than in the traditional approach.

Our future works include a large-scale evaluation under realistic traffic settings of the routing performances of our approach. We will explore the benefits one can expect in terms of both memory and latency from the content-aware forwarding process in comparison to OSPFN approach.

References

[1] Bittorrent makes twitters server deployment 75x faster. Last Access: May 2013.

[2] Ccnx repository protocols. Last Access: May 2013.

[3] Facebook uses bittorrent, and they love it. Last Access: May 2013.

[4] S. Arianfar, P. Nikander, and J. Ott. On content-centric router design and implications. In *ACM CoNext Workshop ReARCH*, 2010.

[5] M. Bari, S. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu. A survey of naming and routing in information-centric networks. *Communications Magazine, IEEE*, 50:44–53, 2012.

[6] G. Carofiglio, V. Gehlen, and D. Perino. Experimental evaluation of memory management in content-centric networking. In *ICC 2011*, June, 2011.

[7] C. Fricker, P. Robert, J. Roberts, and N. Sbihi. Impact of traffic mix on caching performance in a content-centric network. In *IEEE INFOCOM NOMEN Workshops*, 2012.

[8] M. Goldmann and G. Kreitz. Measurements on the spotify peer-assisted music-on-demand streaming system. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 206–211, 2011.

[9] F. Guillemin, T. Houdoin, and S. Moteau. Statistics of youtube traffic in orange ip networks. Orange Labs draft, 2012.

[10] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. A performance study of bittorrent-like peer-to-peer systems. *Selected Areas in Communications, IEEE Journal on*, 25(1):155 –169, jan. 2007.

[11] H. Hwang, S. Ata, and M. Murata. Realization of name lookup table in routers towards content-centric networks. In *Proc. of CNSM*, 2011.

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

[13] G. Kreitz and F. Niemela. Spotify – large scale, low latency, p2p music-on-demand streaming. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–10, 2010.

[14] L. Muscariello, G. Carofiglio, and M. Gallo. Bandwidth and storage sharing performance in information centric networking. In *Proc. of the ACM SIGCOMM workshop ICN*, 2011.

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

[16] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 367–378, New York, NY, USA, 2004. ACM.

[17] D. Rossi and G. Rossini. On sizing ccn content stores by exploiting topological information. In *Proc. of IEEE Infocom NOMEN Workshop*, 2012.

[18] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang. Ospfn: An ospf based routing protocol for named data networking. 2012.

[19] M. Wittie, V. Pejovic, L. Deek, K. Almeroth, and B. Zhao. Exploiting locality of interest in online social networks. In *ACM CoNEXT 2010*, Novemver 2010.

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

[21] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, K. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, and E. Yeh. Named data networking (NDN) project. October 2010.