Realistic Storage of Pending Requests in Content-Centric Network Routers

Wei You, Bertrand Mathieu, Patrick Truong, Jean-François Peltier
Orange Labs
Lannion, France
{wei.you, bertrand2.mathieu, patrick.truong, jeanfrancois.peltier}@orange.com

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

***Abstract*—Content-Centric Networking (CCN) is a novel network paradigm, which aims at moving from the traditional host-to-host network model to a client-to-content one. This shift brings many benefits but also leads a big challenge on the hardware technologies for implementing CCN nodes. For example the Pending Interest Table, one critical component in a CCN node, is involved in every CCN message forwarding process, and needs to be updated every time a packet comes in. The implementation of this table requires a memory that is fast for short access time and also large enough for storing the pending Interests. Unfortunately today's memory technologies cannot meet these requirements with the current hash-based architecture. After highlighting this limitation, we present a distributed PIT architecture which is based on the Bloom filter structure. The evaluations validate that our solution can reduce the PIT table size and support higher packet arrival rate. Our solution allows to implement this component on today's fast memory like SRAM. Therefore this proposal can improve the content fetching performance and improve the quality of content delivered in CCN network.**

I. Introduction

The host-to-host communication paradigm that has ruled the Internet for 40 years is now challenged by new usages: people care more about *which* content or information they are really interested in, and less on *where* the information is. Content-Centric Networking (CCN) [6] proposes a new networking paradigm based on a client-to-content model. In CCN, the networking targets are no longer the identified hosts, but the named contents. Each content object is identified by a unique name, which is independent from its location. This promising networking approach requires however revisiting the architecture of routers, which are called *CCN nodes*.

A CCN node has an extended set of features. It has to deal with the *Interest* packets that are emitted by the end-users requesting a content. Since the same content can be requested by many users, CCN nodes have a cache ability (component named *Content Store* in CCN). If the content is not cached, the Interest packet is forwarded to all the potential content holders. The content response (*a.k.a Data* packet) is sent back to all the clients having expressed interest for this content. Thus, a CCN node contains a *Pending Interest Table* (PIT), which maintains the list of received Interests and a *Forwarding Information Base* (FIB), which is used to forward the incoming Interest packets. The FIB table is similar to the FIB table in a traditional IP network but the PIT is a new concept.

Technology	Access time (ns)	Cost ($/MB)	Max. size
SRAM (on-chip)	1	27	$\approx$ 50 Mbits
SRAM (off-chip)	4	27	$\approx$ 250 Mbits
RLDRAM	15	0.27	$\approx$ 2 Gb
DRAM	55	0.016	$\approx$ 10 GB

TABLE I
Current memory technology [8]

Today's hardware, especially memory chips, do not enable the development of all features of CCN nodes. If we look at memory technologies in Tab I, we observe that the chips with the shorter access time have small storage, while the chips that can offer a larger space do not have a fast access time. These limitations, which were highlighted in [8], call for the development of new techniques in order to fit next-generation networking proposal into today's memory technologies.

In this paper we focus on the PIT table, which has not received much research attention so far. Previous works have mainly focused on the Content Store [2, 7] or on memory architecture for efficient FIB forwarding processes [5]. The PIT needs to memorize every different incoming Interest, and to remove the Interest entry when the associated content comes back. Since almost every incoming message could lead to an update process in this table, the PIT should be built on a fast memory chip. On the other hand, the PIT should have a large storing capacity because the length of CCN content name is at least as large as the length of an IPv6 address

We show in §III that current PIT implementation, which is based on a hash-table, cannot fit with today's memory technologies. We present in §IV the new PIT implementation that we have introduced in [10]. This proposal is called **DiPIT** as it is based on a *Distributed* PIT table. In §V, we analyze the feasibility and the scalability of DiPIT in terms of memory space consumption and networking load aspect. Finally, we describe in §VI the deployment of DiPIT into a hierarchical CCN network, then we make recommendations for both the choice of a PIT implementation and the memory technologies, based on the expected interest rate in the network.

II. Preamble: Modeling PIT table

We provide in the following a model for the CCN PIT table. We also introduce the main notations that we used throughout this paper in Table II.

Parameters	Meaning
n_{face}	number of interfaces in the CCN node
λ_i	Interest packet arrival rate at the interface i
λ_{pit}	Interest packet entering rate to the PIT table
RTT	average Data packet round-trip time
CS_{hit}	hit ratio of the Content Store
$\tau_{interest}$	ratio of Interest packets for the same ContentName. It depends on the traffic distribution
$Nbr_{pitEntry}$	number of stored entries in the PIT table

TABLE II
CCN NODE MODELING PARAMETERS

When an Interest arrives a CCN node, it should first be checked in Content Store. This Interest packet is then considered by the PIT table if no Data cached in the Content Store can fulfill this request. Thus the Interest entering rate for the PIT is:

$$\lambda_{pit} = \sum_{i=1}^{nbr_{face}} \lambda_i \cdot (1 - CS_{hit})$$

The entries in PIT table are then "consumed" by the Data packets, so the number of entries is in average $\lambda_{pit} \cdot RTT$. We also consider redundant Interest packet, thus we obtain that the actual number of stored PIT entries is:

$$Nbr_{pitEntry} = \sum_{i=1}^{n_{face}} \lambda_i \cdot (1 - CS_{hit}) \cdot RTT \cdot \tau_{interest} \quad (1)$$

III. Analysis on the Centralized Hash Table

The PIT table described in the seminal CCN paper [6] is a centralized hash table. This implementation is also chosen by default in the open-source release CCNx [1]. We analyze in this Section the scalability of the hash-based CCN PIT table. In our evaluation we consider only the two fastest memory technologies, the SRAM and the RLDRAM. The SRAM could be built either on-chip or off-chip (Table I). The on-chip SRAM memory offers a faster access time however it is limited at 50 Mbits size. This size is too small for the PIT table, so SRAM refers to the off-chip SRAM in the rest of the paper.

We highlight in Equation (1) that the number of PIT entries depends on both CS hit ratio (CS_{hit}) and traffic popularity distribution ($\tau_{interest}$). Both parameters do not vary much for a given CCN node, and independent of the implementation design of the PIT. To simplify the analysis, we set $CS_{hit} = 0$ and $\tau_{interest} = 1$.

A. Table size and Cost

Each entry in a hash table PIT contains the key value of ContentName (of length $size_{key}$) and the incoming interface identifier information (of length $size_{face\_id}$). The estimated hash table size is then:

$$Size_{hash} = (size_{key} + size_{face\_id}) \cdot \sum_{i=1}^{n_{face}} \lambda_i \cdot RTT \quad (2)$$

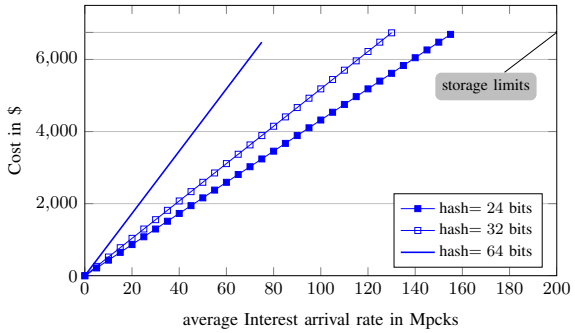

Fig. 1. Required SRAM memory cost vs. λ_{in}

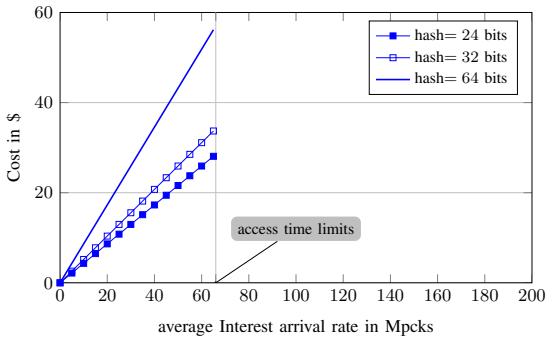

Fig. 2. Required RLDRAM memory cost vs. λ_{in}

In our evaluation, we suppose a router with $n_{face} = 4$ interfaces. The parameter λ_{in} ranges from 0 to 200 Millions of packets per second (Mpcks). The average packet RTT is 80 ms as defined in [8]. In order to hash the content names, we use three different hash values: 24 bits, 32 bits and 64 bits.

In Figure 1 and Figure 2 we present the cost of implementing a hash-based centralized PIT table on a SRAM memory or on a RLDRAM memory, respectively. We can see that in Figure 1, all the three curves stop at $cost = \$6{,}750$, which refers to the maximum size of SRAM (250 MBytes). We highlight here that a SRAM is fast enough for the processing of packets, but it cannot store the entries for high Interest arrival rate. Even for short hash value like 24 bits, the SRAM can only support up to 150 Mpcks packet arrival rate. The three curves in Figure 2 stop at $\lambda = 66$ Mpcks since the RLDRAM cannot process more packets per second. If the size is no more a problem, RLDRAM is too slow for most CCN nodes.

To summarize this first evaluation, only SRAM with hash values lower to 32 bits can meet both packet arrival rate and memory size requirements of a high-level CCN node.

B. Collision ratio

Short hash value length enables the implementation of PIT with large storage of Interests. But the shorter is the hash value length, the more probable are collisions. A collision in hash table occurs when distinct elements are coded at the same entry of a hash table. The elements that suffer from a collision are stored in a linked list at the same hash entry, therefore a collision does not cause packet loss (except if the

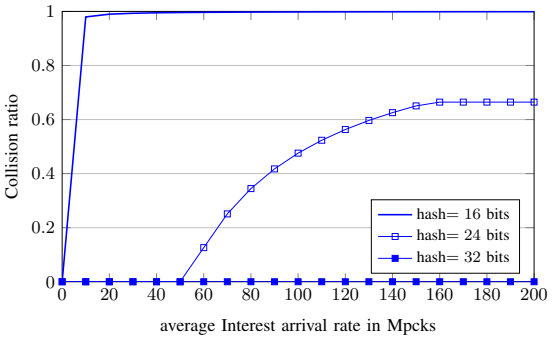

Fig. 3. Predicted collision ratio of SRAM in function of λ_{in}

memory is full), but information retrieval is much longer. Here we evaluate the collision rate for different hash value lengths.

$$Collision = \begin{cases} \dfrac{nbr_{in} - 2^{size_{key}}}{nbr_{in}}, & nbr_{in} > 2^{size_{key}} \\ 0 & nbr_{in} \leq 2^{size_{key}} \end{cases}$$

$$\text{where } nbr_{in} = \sum_{i=1}^{n_{face}} \lambda_i \cdot RTT \tag{3}$$

We then evaluate the collision ratio for hash value lengths 16 bits, 24 bits and 32 bits (since we observed that higher values are not possible). In Figure 3, both 16 bits and 24 bits experience collisions for small Interest arrival rate. Moreover collision ratios quickly reach their maximum level. Only the 32 bits hash values present low collision rate until 200 Mpcks.

C. Discussion

Through both evaluations in §III-A and §III-B we observe that the selection of the memory technology depends on the packet arrival rate. For example, a RLDRAM with a 32 bits (or more) hash value can support an edge router with a low packet arrival rate (e.g. less than 66 Mpcks). However, for a bigger router (e.g., a core network router), the packet rate is generally higher than 66 Mpcks and a SRAM with a 32 bits hash value is the only option. If the rate is higher than 130 Mpcks, the hash table implementation has some critical limitations because neither SRAM nor RLDRAM can support both memory and access time requirements. Alternative solutions should be designed.

IV. DiPIT: A distributed Bloom filter based PIT

Since access time of off-chip SRAM is not the bottleneck, the key issue is the reduction of the table size. In a recent paper [10], we proposed a distributed CCN PIT table system, namely DiPIT. In this paper, we consider DiPIT as an alternative PIT implementation. We give a brief description of DiPIT hereafter. More details can be found in [10].

In DiPIT, the single centralized PIT table is distributed on *each* router interface. The distributed small PIT table associated to each interface is named a *PITi*. Each PITi is constructed with a counting Bloom filter. One PITi is in charge of the incoming Interest messages and Data messages only through the interface it is related to.

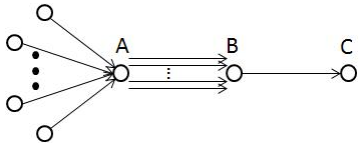

Fig. 4. Duplicated Interest coming through different interfaces

When an Interest message arrives on an interface, the message is checked in the related PITi. If there is already an entry for this Interest, this packet is not forwarded upstream. Otherwise, the Interest message is coded in the counting Bloom filter, and forwarded upstream according to the FIB table. An incoming Data packet is checked with *all* PITi tables. The Data is forwarded through any interface whose PITi returns a positive verification result. The related footprints are then deleted from the counting Bloom filters.

The Bloom filter can introduce false positives. The false positive ratio can be calculated as :

$$fp = (1 - e^{\frac{-n \cdot k}{m}})^k \tag{4}$$

where n is the number of inserted elements, k the number of hash functions, and m the length of the vector. In our case, n is the product of the packet rate λ and the RTT.

In the design of CCN, if one node receives multiple Interests on the same content, only the first one is forwarded. Our DiPIT architecture can avoid sending the duplicated Interest coming through the same interface, but not for those coming from other interfaces, because every PITi is independent from each other. This leads to an extra networking load. However, such event can only occur when two Interest packets arrive on distinct interfaces in a time frame of RTT (otherwise the second Interest packet would find the Data in the Content Store). We analyse now this potential limitations. Let us suppose the worst case, i.e. several interfaces receiving the same Interest within a Data round-trip time. No matter how many interfaces of the CCN node receive and forward the same Interest, the impact on the whole network is only within one hop. The case is illustrated in Figure 4. The arrows denote Interest packet propagation. Node A receives the same Interest several times from its different interfaces. According to its FIB table, all these Interests will be forwarded to node B. For node B, all these Interests come from the link between B and A via the same interface. Thus only the first incoming Interest is further forwarded to the next hop C. The other Interests are blocked by the PITi tables because they arise a positive matching in the Bloom filter.

V. Evaluations of the distributed PIT

We now evaluate the cost and performance aspects of DiPIT. We also evaluate whether DiPIT with acceptable false positive ratio can allow the SRAM technologies to meet both memory and access time requirements. This analysis has not been presented in [10].

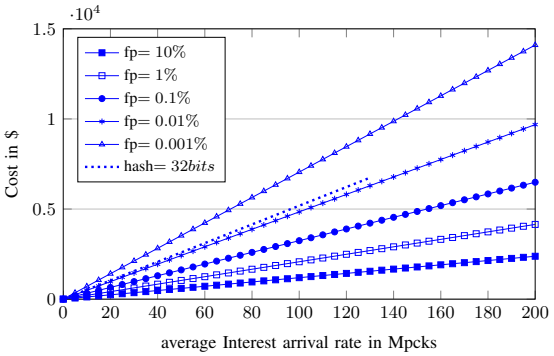

Fig. 5. PITi – Required memory size vs. λ_{in}

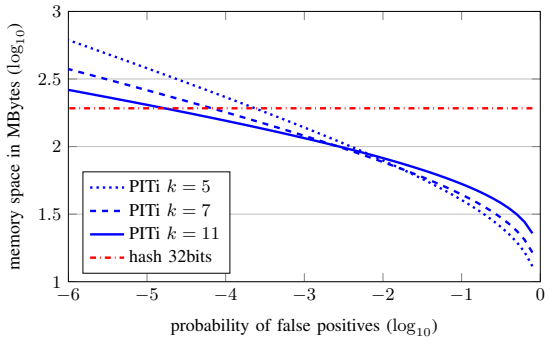

Fig. 6. Required memory size vs. false positive

A. Table size and cost

The DiPIT table size depends on the length of the Bloom filter and the number of interfaces n_{face}. The length of a Bloom filter m can be calculated from the acceptable false positive fp, the number of applied hash functions k, and the number of inserted elements n. From Equation (4), we have

$$m = \frac{-n \cdot k}{log(1 - fp^{\frac{1}{k}})}$$

Each required memory size for a counting Bloom filter is the product of the vector length and the counter size. We denote the counter size as $Counter$. Thus the estimated memory size of the entire DiPIT system is:

$$Size = Counter \cdot \sum_{i=1}^{n_{face}} \frac{-\lambda_i \cdot RTT \cdot k}{log(1 - fp^{-k})} \qquad (5)$$

To evaluate the performance of DiPIT, we consider the following settings. We use SRAM because RDLRAM does not support high packet arrival rate. Again λ_{in} ranges from 0 to 200, $RTT = 80$ ms, and the CCN node has four interfaces. We give 2 bits for each counter. Since today's Internet services tend to tolerate a packet loss rate of 1% at least, we thus set our acceptable false positive in the range from 0.001% (which is actually very low) to 10%. In most previous works, the acceptable false positive probability is comprised between 0.01% and 10% [3, 4, 9, 11].

In Figure 5, we present the cost for DiPIT and different acceptable false positive parameters. To ease the comparison to hash-table based solutions, we also show the cost for a hash table with 32 bits hash value (the only one that is acceptable according to our analyse). Out of all DiPIT solutions, the only one that a bigger cost is when the acceptable false positive rate is 0.001%. Moreover, DiPIT can tolerate larger packet arrival rates. The DiPIT solution can overcome the SRAM memory space limitation because the whole PIT table is now distributed to each interface and each PITi table can be independently built on a SRAM chip. Thus even if the size of the overall PIT is above the 250 MBytes limitation, the size of each PITi is smaller than the maximum memory size.

B. Table size and false positive

The acceptable false positive ratio is a key parameter of the DiPIT system because the memory size derives from the false positive ratio in Equation (5). In the evaluation we analyse the impact of false positive on the table size. We fix the average Interest arrival rate at 100 Mpcks and the Data RTT as 80 ms. In Figure 6 the variable k is the number of hash functions used in DiPIT. The required memory space is calculated according to different false positive and hash functions number. We can see that the 32 bits hash table overcomes the DiPIT when the acceptable false positive is lesser than 0.03% ($k = 5$, -3.5 on the x-axis) or 0.003% ($k = 11$,-4.5 on the x-axis).

C. Extra Data traffic load

Because a PITi produces a false positive, some Data might be sent out although no Interest matches. Here we analyse the generated extra networking load. Let us suppose a CCN node with n_{face} interfaces receives nbr_{data}. We denote by fp_{j_i} the false positive ratio of the interface i at the node j. We set N_{hop} as the number of times the "false" Data can be forwarded. The sum of the "false" Data packets that are sent out by one node impacts the N_{hop} networking link such as:

$$Number_{data} = \sum_{j=1}^{N_{hop}} (nbr_{data} \cdot \sum_{1=1}^{n_{face}} fp_{j\_i}) \qquad (6)$$

We fix the average Data arrival rate at 100 Mpcks and the Data RTT as 80 ms. The acceptable false positive is 1%, 0.1% and 0.01%. From Figure 7 we can see that after the second hop, the extra traffic is nearly zero. In fact, one node can generate some extra Data packets but this extra traffic is stopped at the next hops because the probability several (even only 2) nodes produce the same mistake is very low.

D. Discussion

These three evaluations prove that the DiPIT can overcome the limitations of fast memory chips. When the packet arrival rate exceeds 130 Mpcks, only the distributed system can allow SRAM technologies to be used for PIT. The trade-off is the false positive. DiPIT is especially the solution for high-level networking routers (like a core router or the peering router) that have a higher packet arrival rate. And since DiPIT requires

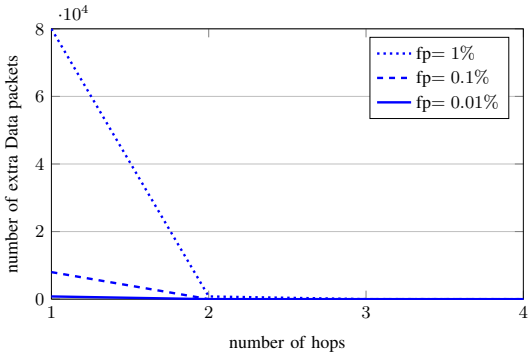

Fig. 7. Extra Data packet load decreased during propagating

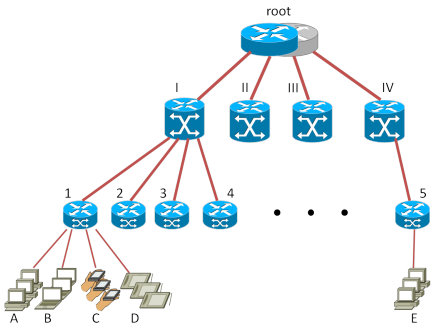

Fig. 8. A hierarchical networking topology

less memory than a hash table, DiPIT can also be considered for CCN nodes with smaller packet rate arrival (if this node is not too sensitive to false positive effects). In Bloom-filter based PIT system more hash functions can reduce the filter size (Figure 6). If we seek a faster performance and we are less sensitive to the cost, we can use less hash functions (as $k = 3$ or 5) and larger filter in order to reach a targeted acceptable false positive. On the contrary, if we want to limit the cost, we can implement more hash functions (as $k = 7$ or 11) in order to reduce the memory space.

VI. Case Study: Hierarchical Network

We now present a case study on a hierarchical network. We especially identify the conditions under which a traditional hash table can be implemented and we show that DiPIT can significantly expand the applicability of CCN in networks with today's memory technologies.

A. Analysis

We consider a three level hierarchical topology, illustrated in Figure 8. We denote the peering router as $root$. The $root$ is connected with four core routers, which are denoted as I, II, III and IV. Each core router is connected to four edge routers, noted as 1, 2, 3, 4. The set of end-users which is connected to an edge router is noted as $A, B, \ldots, E$. Each set of end-users generates Interest packets with λ_{in} Mpcks. Each internal link has a d ms delay. The $root$ has D ms delay with outside network. The acceptable false positive for the $root$, the core router and the edge routers are denoted fp_{root}, fp_I, and fp_1 respectively. The number of hash functions for the Bloom filter are k_{root}, k_I and k_1 respectively. The Content Store hit ratio for each level router is CS_{root}, CS_I and CS_1 respectively. The ratio of identical Interest packets among the traffic is $\tau_{interest}\%$.

The edge router 1 receives the Interest messages from four interfaces at a rate λ_{in} Mpcks. The RTT for each interface is $2 * d + D$ ms. We suppose that $\tau_{interest}$ are identical for the traffic of every interface and every router. Thus the overall distributed PIT size for an edge server is:

$$Size_1 = 2 * 4 * (1 - CS_1) * \tau_{interest} * \frac{-\lambda_{in} * 10^6 * (2 * d + D) * 10^{-3} * k_1}{log(1 - fp_1^{-k_1})} \quad (7)$$

The core router I receives the Interest packets from four edge routers, so it has a higher input packet rate (when the Content Store has a realistic hit ratio). The Interest arrival rate at each interface is equal to the Interest leaving rate at the edge router up-streaming interface. Thus the overall distributed PIT size for the core router is:

$$Size_I = 2 * 4 * (1 - CS_1) * (1 - CS_I)(\tau_{interest})^2 * \frac{-\lambda_{in} * 10^6 * (d + D) * 10^{-3} * k_I}{log(1 - fp_I^{-k_I})} \quad (8)$$

We can similarly derive the overall DiPIT size for the $root$ as:

$$Size_{root} = 2 * 4 * (1 - CS_1) * (1 - CS_I) * (1 - CS_{root}) (\tau_{interest})^3 * \frac{-\lambda_{in} * 10^6 * D * 10^{-3} * k_{root}}{log(1 - fp_{root}^{-k_{root}})} \quad (9)$$

B. Settings

We now build the evaluation system. We consider that edge router receives Interest packets from four set of terminals at rate 10 Mpcks and $\tau_{interest}\% = 95\%$. The transmission time on each internal link is 20 ms, so RTT is smaller or equal to 80 ms. We do not consider Content Store caching because our main motivation is to identify the right technologies for PIT, so the impact of $CS_{hit} = 0$ is identical on every technology. Since we showed that false positive has not a significant impact on the whole network, we set a relatively high acceptable false positive ratio for the root router (1%) while we use 0.1% for the edge and core routers. Finally, we used 7 hash functions for the edge routers, because the packet rate is not high. On the contrary, core routers and root routers need faster performance, thus we use 5 and 3 hash function for them, respectively.

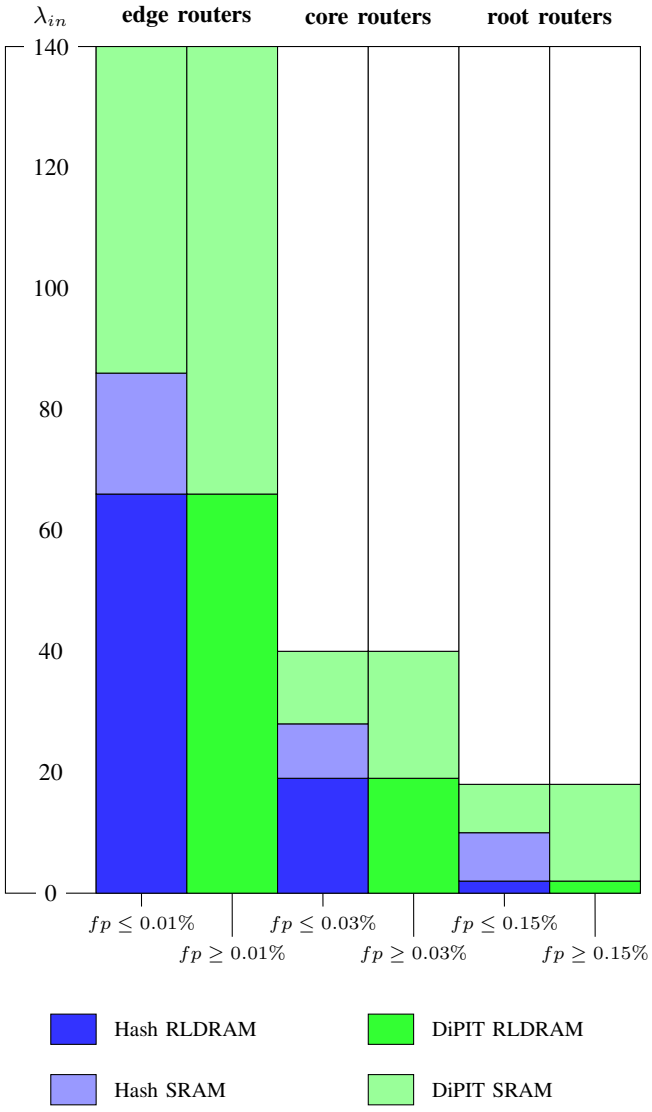

Fig. 9. Summary of the best choices for PIT table, according to the cost

C. Discussion

We now summarize how to choose the PIT architecture (traditional hash table or DiPIT) and the cheapest memory technology for the routers at different levels. See Figure 9. In short, all green blocks correspond to configurations where DiPIT allows an implementation of PIT table in today's memory technologies although it is impossible to make it with traditional hash tables (the blue blocks).

We now explain how to read Figuree 9. We take the edge router as an example. If we can accept a false positive ratio that is larger than 0.01% (right column), DiPIT is always cheaper than centralized hash tables. And if the λ_{in} is smaller than 66 Mpcks (calculated with $fp = 0.1\%$), it is better to implement with RLDRAM because it is cheaper. If we are sensitive to the false positive ($fp < 0.01\%$), the hash table is a better solution. However when the λ_{in} exceeds 86 Mpcks, the hash table can no more be used. Then DiPIT with SRAM is the only option until $\lambda_{in} = 140$ Mpcks. Recommendations for core and root routershave to be read the same way.

Figure 9 highlights the main contribution of this paper: we provide a way to let an operator make choices for the deployment of a CCN network. Consider for example an operator with incoming traffic at 15 Mpcks, and the requirement to limit the false positive for the edge, core routers, and root router at 0.02%, 0.02%, and 0.2% respectively. The right choice for the edge routers is DiPIT+RLDRAM, for the core routers is Hash+RLDRAM and for the root router is DiPIT+SRAM.

VII. Conclusion

The CCN paradigm can bring many benefits to Internet evolution. However today's memory technologies are not ready to support this innovation in term of memory space and access time, especially for the PIT table. In this paper we analysed the feasibility and the scalability of the hash table. We also analyzed the DiPIT system, which aims at being implemented in smaller so faster memory chips. Each technique has its advantage. The hash table presents some limitations at the table size and the performance speed aspects but it does not produce any extra network load. The DiPIT system requires less memory space and enables implementations on fast memory chips, so it can handle higher packet arrival rate. However DiPIT introduces some extra network load because Bloom Filter can experience some false positive. Different networking conditions call for different approaches. We also showed that mixing different memory chip technologies is possible.

In future works, we will deploy modified version of CCNx on a large testbeds for further analysis. We also consider the study of a distributed table for FIB and Content Store. Our goal is to make sure that CCN protocols can be implemented in off-the-shelf routers.

Acknowledgment

This work is partially funded by the French National Research Agency (ANR), CONNECT project.

References

[1] Project CCNx. http://www.ccnx.org/.
[2] S. Arianfar, P. Nikander, and J. Ott. On content-centric router design and implications. In *ACM CoNext Workshop ReARCH*, 2010.
[3] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.
[4] D. Guo, J. Wu, H. Chen, and X. Luo. Theory and network applications of dynamic bloom filters. In *IEEE INFOCOM*, 2006.
[5] H. Hwang, S. Ata, and M. Murata. Realization of name lookup table in routers towards content-centric networks. In *Proc. of CNSM*, 2011.
[6] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. In *ACM CoNEXT*, 2009.
[7] 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.
[8] D. Perino and M. Varvello. A reality check for content contric networking. In *ACM Sigcomm Workshop ICN*, 2011.
[9] Y. Qiao, T. Li, and S. Chen. One memory access bloom filters and their generalization. In *IEEE INFOCOM*, 2011.
[10] W. You, B. Mathieu, P. Truong, J.-F. Peltier, and G. Simon. DiPIT: a distributed Bloom-Filter based PIT table for CCN nodes. In *ICCCN (to apear)*, 2012.
[11] M. Yu, A. Fabrikant, and J. Rexford. Buffalo: bloom filter forwarding architecture for large organizations. In *ACM CoNEXT*, 2009.