Supporting the Web with an Information Centric Network that Routes by Name

A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano

CNIT - Department of Electronic Engineering, University of Rome "Tor Vergata"
Via del Politecnico 1, Rome (Italy)
{andrea.detti, matteo.pomposini, blefari, stefano.salsano}@uniroma2.it

Abstract - Information Centric Networking (ICN) is a new paradigm in which the network layer provides users with content, instead of providing communication channels between hosts, and is aware of the name (or identifiers) of the contents. A fundamental ICN operation is the routing of content requests towards a node that is able to provide the requested content. To meet this goal, different routing architectures have been proposed so far.

In this paper, we consider a network that uses a *routing-by-name* architecture, i.e. content requests are routed on the base of the content name by using a *name-based routing table*. We focus on the scenario of fetching Web contents, assuming to use ICN in place of traditional TCP/IP means. In this scenario we need to handle tens of billions of name-based routes, due to the high numbers of Web contents and to the limited aggregability of their names. Consequently, re-using the existing architecture of an IP router would result in two severe problems. First, the current Forwarding Information Base (FIB) technology is unable to contain all name-based routes. Second, implementing a so large Routing Information Base (RIB) requires a very costly hardware. In order to overcome these problems, we propose a routing-by-name architecture, named Lookup-and-Cache, where the FIB is used as a *cache of routes*, while the RIB is stored in a remote and centralized Routing Engine. By analyzing real Internet traces, we prove the effectiveness of the proposed architecture, which we also show to be feasible with current technology. In fact, our ICN nodes require to have only a limited set of routes in their FIB, even when supporting a high number of traffic flows.

We have implemented our proposed Lookup-and-Cache solution within the CCNx software framework and we used this implementation to assess system performance, such as download delay, lookup rate and fairness.

The paper is completed with a discussion on how ICN can be used not only to fetch Web contents but also for other scenarios.

Keywords - Internet Architecture, Future Internet, Information-Centric Networking, Routing, Caching, Scalability

1. Introduction

Information Centric Networking (ICN) is a concept proposed some time ago under different names [1][2], which is attracting more and more interest, recently (see e.g. the papers [3][4][5][6], the workshop [36] and the projects [6][7][8][9][10][11]). ICN proposes a shift from the traditional host-to-host communication to a content-to-user paradigm, which focuses on the delivery of the desired content to the intended users. The basic functions of an ICN infrastructure are to: i) address contents, adopting an addressing scheme based on names (identifiers), which do not include references to their location; ii) route a user request, which includes a "destination" content-name, toward the "closest" copy of the content with such a name; iii) deliver the content back to the requesting host.

ICN basic concepts were first proposed in [1], with the so-called TRIAD architecture. TRIAD is an overlay network of content-routers. Content routers *route-by-name* content requests, i.e. they route content requests on the basis of the content-name towards the "best" suitable server. Once the server is reached, the content is delivered to the user by means of a plain TCP/IP session.

A second seminal paper is [2], which proposed the so-called DONA architecture. DONA introduced important issues, such as security and naming, and proposed to use self-certifying-names. Similarly to TRIAD, DONA uses an overlay network to route-by-name content requests towards the best server, and then delivers contents via TCP/IP. Therefore, the route-by-name procedures are involved only at the start of a download session and the network path followed by a content request could be different from the one used to deliver the data using TCP/IP.

The third seminal paper on ICN is [3], which proposed a so-called content-centric network (CCN). CCN does not rely on an underlying IP infrastructure. Thus, CCN could completely replace the IP layer. CCN procedures route-by-name requests for *chunks* of contents. Furthermore, CCN mechanisms also handle the delivery of chunks to the user, rather than delegate this operation to TCP/IP. To download the whole required content, a user sequentially downloads all its chunks. Hence, route-by-name procedures are

continuously involved during a content download and require line-rate processing speed. The path selected to route-by-name a chunk request is also used in the reverse direction to deliver the corresponding chunk data. The matching between the request path and the data path facilitates the exploitation of *en-route* (or *in-network*) caches within network nodes [14]. CCN attracted a significant number of researchers to the ICN field, also thanks to the development and release of an open source implementation of the main CCN concepts, called CCNx [17]. In the following sub-section we report our view on pros and cons of ICN.

1.1 Pros and cons of ICN

In our view, an ICN would offer the following advantages:

Efficient routing by name; even though Content Delivery Networks (CDNs) offer efficient mechanisms to route contents, they cannot use network resources in an optimal way because they operate over-the-top, i.e. without knowledge of the underlying network topology. ICN would let ISPs perform native content routing with improved reliability and scalability of content access. This would be a built-in facility of the network, unlike today's CDNs.

In-network caching; caching enabled today by off-the-shelf HTTP transparent proxies requires performing stateful operations, because the delivery of cached content is based on the connection-oriented TCP. The burden of a stateful processing makes it very expensive to deploy caches in nodes that handle a large number of user sessions. ICN would significantly improve efficiency, reliability and scalability of caching, with interesting applications especially for video.

Simplified support for peer-to-peer like communication, without the need of overlay dedicated systems. Users could obtain desired contents from other users (or from caching nodes) thanks to content-routing functionality, as it is done today with specialized applications, which, once again, do not have a full knowledge of the network and involve only a subset of possible users.

Per-content quality of service differentiation, providing different performance in terms of both transmission and caching. Network operators (especially mobile ones) are already trying to differentiate the quality and the priority of content, but they are forced to resort to complex and hardly scalable deep packet inspection technologies. ICN would let network operators differentiate the quality perceived by different services without complex, high-layer procedures, and off-load their networks via caching, a very handy functionality, particularly for mobile operators who can differentiate the quality and the priority of content transferred over the precious radio real estate.

Handling of mobile and multicast communications, simplifying handovers and stateful nodes. As regards handovers, when a user changes point of attachment to the network, she will simply ask the next chunk of the content she is interested in, without the need of storing states; the next chunk could be provided by a different node than the one that it would have been used before the handover. Similar considerations apply for multicasting. Several users can request the same content and the network will provide the service, exploiting caches, without the need of overlay mechanisms;

Content-oriented security model; securing the content itself, instead of securing the communications channels allows for a stronger, more flexible and customizable protection of content and of user privacy. In today's network contents are protected by securing the channel (connection-based security) or the applications (application-based security). ICN would protect information at the source in a more flexible and robust way than delegating this function to the channel or the applications [4]. In addition, this is a necessary requirement for an ICN: in-network caching requires to embed security information in the content data-unit, because content may arrive from any node and we cannot trust the serving node; thus, end-users must be able to verify the validity of the received data and caching nodes must do the same to avoid caching fake contents.

Support for time/space-decoupled model of communications, simplifying implementations of publish/subscribe service models, and allowing "pieces" of network, or sets of devices to operate even

when disconnected from the main Internet (e.g. sensors networks, ad-hoc networks, vehicle networks, delay-tolerant-networks, social gatherings, mobile networks on board vehicles, trains, planes).

On the cons side, ICN has some drawbacks and challenges. A first, obvious, con is that it requires **changes in the basic network operation**, which per se is already a big obstacle to take-up of this approach. A second con is that it raises **scalability** concerns: i) the number of different contents and corresponding names is much bigger than the number of host addresses; this has implications on the size of routing tables and on the complexity of lookup functions; ii) in some proposed ICN architectures [3], delivering contents back to requesting users requires maintaining states in network nodes.

In the following subsections we start discussing the general design issues of an ICN and then we present the specific contributions of this paper.

1.2 General design issues

The design of an ICN brings up several issues (see e.g. [19]) such as:

Primitives & interfaces, which define the relationship of the ICN protocols with the overall architecture.

Naming scheme, which specifies the identifiers for the contents addressed by ICN.

Route-by-name mechanism, used by ICN nodes to relay an incoming content request (aka Interest messages) to an output interface.

Routing protocols, used to disseminate information about location of contents, so as to properly setup the name-based routing (aka, RIB) and forwarding tables (aka FIBs).

Data forwarding mechanisms, that send back the content to the device that issued the related content request. It is not possible to assume that data forwarding can use the route-by-name mechanisms, since, typically, devices/interfaces are not addressed by the routing plane of an ICN.

Cache replacement algorithms, which define the policy to replace the contents of a cache, also considering that each single cache is part of a network of caches [14].

Segmentation & transport mechanisms (see e.g. [20]), needed to split a content in different chunks, ensure a reliable transfer of chunks from the origin node (or from a cache node) towards the requesting node and counteract congestion.

Security & privacy issues, which tackle (at least) three aspects: 1) how to guarantee content authenticity and how to protect the network from fake content, which could also pollute network caches; 2) how to guarantee that content be accessed only by intended end users, and 3) how to protect information consumers from profiling or censorship of their requests.

1.3 Main contributions and structure of the paper

In this paper we restrict our attention on the routing-by-name issue. Furthermore, we consider a scenario where the ICN is used to fetch Web contents. The rationale underlying the choice of this "focused" scenario is twofold: first, we believe that the fetching of Web contents is a relevant "use-case" for an ICN; second, in this case we have real traces and data, to assess performance and substantiate our claims.

However, at the end of the paper, we discuss also how ICN can be used not only to fetch Web contents but also for other scenarios.

In our scenario, we need to handle tens of billions of name-based routes, due to the high numbers of Web contents and the limited aggregability of their names. Consequently, if we reused the current architecture of an IP router based on Forwarding Information Bases (FIB) and Routing Information Bases (RIB), we would face two severe problems: first, the current FIB technology is unable to contain all the ICN routes; second, realizing a so large Routing Information Base (RIB) requires a very costly hardware. To overcome these problems, we propose a routing-by-name architecture, named Lookup-and-Cache, where the FIB of a node is used as *cache of routes*; the RIB is stored in a remote and centralized routing engine. Due to the Zipf nature of the statistical distribution of Web contents by their popularity [35], the Lookup-and-Cache architecture is feasible, since the number of routes *concurrently*

needed by a node to forward traffic is rather small and much lower than the capacity provided by current FIB technology.

The structure of the paper and its main contributions are as follows: in Section 2 we define the reference model. In Section 3 we present the understanding of the logical process that produces ICN routes from content-names, with an initial estimation of the number of the ICN routes required to support the Web. In Section 4 we present the understanding of the practical problems to be faced to implement ICN routing-by-name by using current router architectures. In Section 5 we propose our Lookup-and-Cache routing architecture and a possible cache replacement policy. In Section 6 we present the finding that the Zipf nature of the Web results in a limited number of ICN routes that a node should *concurrently* have in its FIB; hence, the route caching is a practical approach and the Lookup-and-Cache architecture can be deployed also with current technology. In Section 7 we describe a real implementation of the Lookup-and-Cache architecture within the CCNx framework and analyze its virtues and limits by means of a testbed. In Section 8 we discuss how ICN can support also scenarios different than the Web contents fetching considered in our paper. In Section 9 we discuss previous related work. We draw our conclusions in Section 10. Some supporting results are reported in Appendices I and II of [47], to limit the length of the paper.

The network model followed in this paper is based on the CCN architecture [3]. However, our findings can be applied also to other solutions, such as our CONET [22], and, more in general, to any ICN that routes-by-name content requests at line-rate. In order to allow reproducibility of our findings, all the software and data sets that we used are available in [37].

2. ICN reference model

The aim of this section is to present the specific ICN network and naming model we refer to (see Fig. 1). We have ICN nodes interconnected by "sub-systems" [22]. Sub-systems use an underlying technology to connect ICN nodes and they can be implemented in several different ways. For instance, a sub-system

could be a public or private IP network, an overlay UDP/IP link, a layer-2 network, a PPP link, etc. This is the same concept used in current IP networks, in which different layer 2 technologies connect IP hosts and routers. Within an ICN sub-system we may have: ICN end-nodes (or clients) that download contents; ICN serving-nodes (or servers) that provide contents and ICN nodes that relay ICN data-units between sub-systems and may cache data.

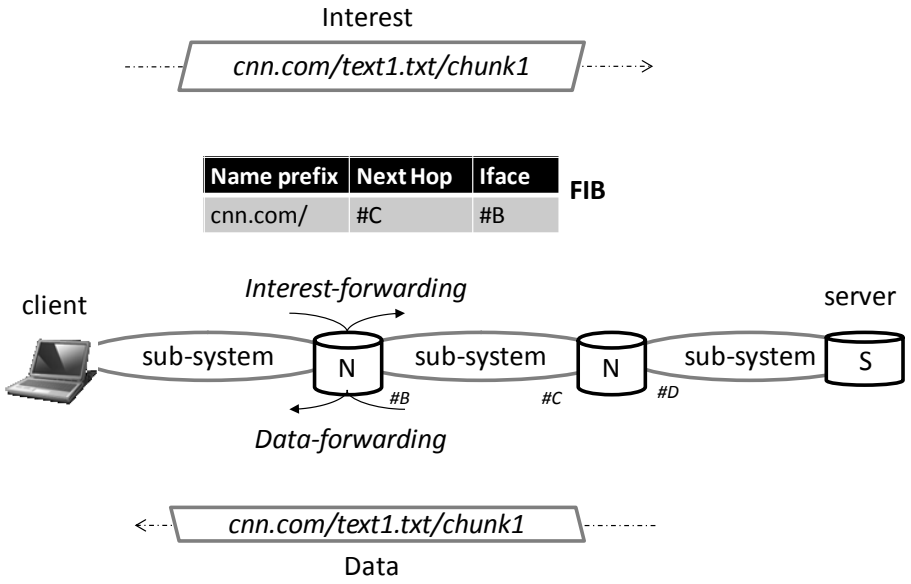

Fig. 1 – Network model

To provide a content, a server splits the content in blocks of data, named *chunks*, and assigns a unique network identifier to each chunks. A network identifier is a string like "cnn.com/text1.txt/chunk1", which is said to be the "name" of the chunk.

The role of the ICN protocols is to discovery and delivery named chunks [21]. In order to fetch a chunk, a user issues a data unit, named *Interest* message, which contains the name of the chunk. ICN nodes *route-by-name* the Interest message, by using a longest prefix matching forwarding strategy and a name-based routing table. We name the entries of the name-based routing table as *ICN routes.* An ICN route has a format like <name-prefix, output port identifier, next hop information>. A name-prefix should be

either the full name of a chunk, e.g. “cnn.com/text1.txt/chunk1”, or a continuous part of it, starting from the first left character e.g. “cnn.com/”.

The first en-route device that has the chunk sends it back within a data unit, named *Data* message, which includes the chunk name. Network nodes forward the Data message towards the requesting client, through the same sequence of ICN nodes previously traversed by the Interest message. The Data forwarding process exploits reverse-path information either temporary stored in the traversed nodes during the Interest forwarding process (see Pending Interest Table of [3]), or contained in the header of Data message, and previously collected in the Interest message during its forwarding process (see reverse-path source-routing in [22]). Therefore, the routing-by-name process does not involve Data messages, but only Interest messages

Downloading a whole content is achieved by sending a *flow* of Interest messages to retrieve all the chunks of the content. The sending rate of Interest message is regulated by a receiver-centric congestion control mechanism [15], which could be based on the same logic used by TCP. Therefore, in our ICN model, we have endpoints that exchange Interest-Data sequences and the message exchange rate is regulated by the receiver. Dually, in TCP/IP the endpoints exchange Segment-Ack sequences and the exchange is regulated by the sender.

Naming scheme – *As r*egards the naming scheme, several proposals (e.g. [2][3][4][19]) agree in adopting a hierarchical naming. In this paper we assume a rather general hierarchical naming scheme where a name is formed by a sequence of *Components*; i.e. a name has the form “Component_1/Component_2/../Component_n”. This scheme supports current Web URL, where the Component_1 is the domain name (e.g., “cnn.com”) and next Components represent the path of the local resource (e.g., /text1.txt). In addition to these Components, which represent the *content-name*, ICN

requires other specific Components, e.g. to represent the chunk number ("/chunk1"), version, etc. [1]. The full sequence of Components is referred to as the *chunk-name*.

3. On the number of ICN routes

In this section we estimate how many routes an ICN node should handle, to properly forward Interest messages. The assessment is carried out by assuming that: i) the ICN will serve Web contents; ii) current Web servers will become ICN servers, iii) the ICN adopts the hierarchical naming scheme previously presented, iv) as a worst case, the node is within the "default-free" zone of the network, i.e. it does not use a default route.

Fig. 2 (right) shows the logical process that produces ICN routes from content-names. For the sake of clarity, Fig. 2 (left) shows the same process in case of IP. Fig. 2 also summarizes the expected number of content-names, name-prefixes and ICN routes, which we will derive later on in this section. The same values are reported for IP, on the base of the BGP analysis provided in [30][32].

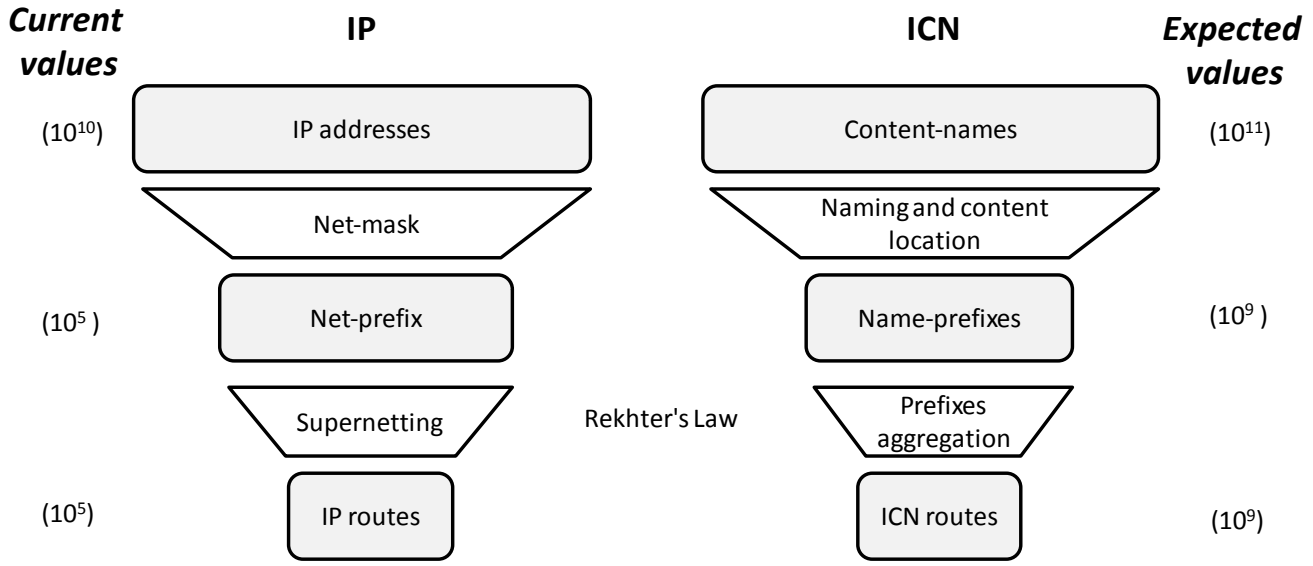

Fig. 2 – From content-names to ICN routes

In what follows, we discuss the right part of Fig. 2, assuming that the left (IP) part is well known.

[1] This hierarchical naming can support also human readable names [3][4] and self-certifying names [2] [19] (in that case, Component_1 is the *Principal* and Component_2 is the *Label*).

At the top of the process we have contents-names. Since we are assuming that the ICN will serve Web contents, we set an initial target of 10^{11} content-names [2]. In facts, the current Web contains a number of resources in the order of 10^{10} [31] and we add a reasonable margin.

The sources of contents are the ICN servers, which advertise content reachability information in the routing plane. For each content, a server could advertise either the full sequence of Components of the content-name, such as "cnn.com/text1.txt" or only a smaller part of it, such as "cnn.com". In the second case, we have an aggregation of routes and we can exploit the longest prefix forwarding strategy. We name the advertised string *name-prefix*, and we say that a name-prefix is for the ICN routing plane what a net-prefix is for the IP routing plane.

The expected number of name-prefixes depends on the number of content-names (10^{11}), on the naming scheme and on the spatial distribution of contents in the network. We use a hierarchical naming scheme, where a content-name is formed by a sequence of Components. On the routing plane, a hierarchical naming scheme enables the aggregation of many content-names in a single name-prefix. For instance, "cnn.com" is a name-prefix that aggregates all the content-names whose first Component is "cnn.com". This kind of aggregation resembles the one obtained by using an IP net-mask.

The effectiveness of the aggregation of content-names in a name-prefix depends on the proximity of contents sharing a common prefix. We are assuming that current Web servers will become ICN servers. Generally, a Web server provides all contents whose URLs have the same domain name, e.g. "cnn.com". Therefore, we assume that an ICN server provides all contents whose names contain the same domain name, i.e. the same Component_1[2]. In this case, ICN servers need to advertise only the first Component of their content names and the number of advertised name-prefixes is equal to the number of Internet domain names. In 2010, the number of second level domains was in the order of 10^8

[2] We note that this assumption does not prevent to support server replication.

[29], so we can set an initial ICN target of 10^9 name-prefixes[3]. This margin also takes into account the fact that some domain names need to be resolved in more than one server address, for redundancy and load sharing purposes (e.g., DNS round robins or HTTP redirects).

The name-prefixes advertised by servers are inserted in the routing tables as ICN routes. During this last step (Fig. 2 right), a further aggregation can take place. This route aggregation resembles the one obtained in IP using the super-netting technique. To achieve an efficient aggregation of routes, the Rekhter's Law [18] states that: "Addressing can follow topology or topology can follow addressing. Choose one". In an ICN, addressing means naming and, hence, the addressing-follows-topology option would imply that content-names contain information about "where" the content is. This clearly contrasts with one of the founding principle of the ICN, therefore the first option of Rekhter's Law is out of question. The second option, "topology-follows-addressing" (used for example by Peer-to-peer DHT-based [42] systems), intrinsically provides a stretch of network paths. The path stretch is critical in the considered ICN network model, where signalling (i.e. content requests) and traffic have to be forwarded on the same path, so as to exploit the caches provided by en-route nodes. To avoid path-stretch, the ICN topology has to follow the topology of the underlying infrastructure, rather than the addressing. Therefore, also the second option of Rekhter's Law is unusable and ICN routing cannot achieve scalability only by using route aggregation.

We conclude that the expected number of ICN routes that a node should handle to forward Interest messages is close to the expected number of name-prefixes advertised by ICN servers, i.e. 10^9. A name-prefix is the first component of a content-name, i.e. the domain name of the server. An ICN route has the form <domain name, output port identifier, next hop information>.

[3] We remark that the value 10^9 is derived by our assumptions on hierarchical naming and on proximity of contents that share a common prefix. Conversely, in case of flat-names or in case of very low proximity, then the number of name-prefixes would be higher and likely close to the number of content-names, i.e. 10^{11}.

We remark that these conclusions are dependent on the "fetching Web contents" scenario, and in particular on the assumptions stated at start of the section. Changing the assumption would change the results. For example, using a "flat" non-hierarchical naming the number of ICN routes would be higher and likely close to the number of content-names, i.e. 10^{11}. If we allow more than one route per name-prefix, e.g. for routing redundancy or multi-homing purposes, the number of ICN routes would be higher than 10^9. Nodes that have a default route, e.g. corresponding to a tier-2 or a tier-3 node of the current Internet, would have a number of ICN routes much lower than 10^9, and so forth.

4. Deploying ICN routing-by-name by using current IP router architectures

The routing-by-name of Interest messages is very similar to the routing of IP packets but, in place of IP-prefixes, the routing-by-name procedure uses name-prefixes. Consequently, it is worth analyzing the feasibility of reusing the architecture of an IP router for an ICN node. Fig. 3 shows a typical architecture of a carrier-grade IP router [26][27] [28]. The router is composed of three major components: one or two routing engines, line cards that host a forwarding engine and a switch fabric. The routing engine handles the routing protocols and stores the routes in a routing table, named Routing Information Base (RIB). In general, the RIB contains several routes to the same destination and it is implemented by means of cheap and slow memories such as DRAM.

The forwarding engine of a line card receives incoming packets and selects the output line card by looking up an on-board routing table, which is named Forwarding Information Base (FIB) [24]. The FIB contains one route per destination, and therefore a smaller number of routes than the RIB. To support packet forwarding at line rate, the forwarding process is carried out by dedicated ASIC chips and the FIB is implemented with fast memories, such as SRAM or TCAM. These memories are costly, consume a lot of power, and do not follow Moore's Law [18]. After the selection of the output interface, the forwarding engine injects the packet in the switching fabric. The switching fabric is an NxN non-blocking crossbar where N is the number of line cards (including the routing engine).

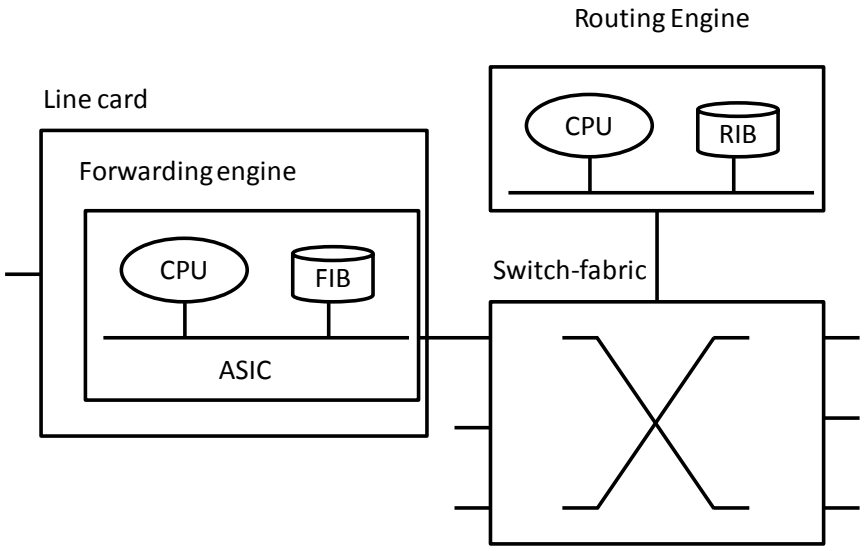

Fig. 3 – Typical IP carrier-grade router architecture

If we want to reuse this architecture to route-by-name ICN Interest messages, we should store ICN routes in the FIB and RIB, and properly update the routing and forwarding logics. Hence, a fundamental check is to verify the practical feasibility of storing all required routes in a FIB and in a RIB.

As regards the FIB, the maximum size of a SRAM chip is today 32 MByte [16]. Assuming that an ICN routing entry is 45 bytes long [4] [2], the number of routing entries storable in a FIB is in the order of 10^6 (i.e. 32MB/45B). In the previous section we estimated that an ICN node should handle 10^9 routes and thus current FIB technology cannot store the whole set of ICN routes.

Let us now analyze the RIB issue. As in IP, the RIB would contain more than one route per name-prefix; this redundancy mainly depends on the peering relationships among Autonomous Systems. For instance, current BGP data obtained from the AS6447 node [30] show that, on average, its RIB contains 31 routes per destination. As a consequence, we assume that the RIB of an ICN node should handle a number of routes in the order to 10^{10}, i.e. one order of magnitude greater than the number of name-prefixes. In this case, the RIB would require hundreds of Gbytes (i.e., 10^{10} *45B) of DRAM memory and a motherboards

[4] 40 bytes are for the name-prefix, 1 byte for the output port identifier and 4 bytes for the next hop info.

with hundreds of memory slots. Current DRAM chips are of 4 GB and motherboards of "expensive" carrier-grade IP routers can host up to 4 memory slots [27][28]. This means that the required increase of capacity is in the order of 10^2. We can conclude that supplying each network node with a motherboard with hundreds of memory slots would dramatically increase the deployment cost of an ICN network, with respect to an IP network.

5. The Lookup-and-Cache routing architecture

In order to cope with the capacity issue of the FIB and with the cost issue of the RIB, we propose a *Lookup-and-Cache* routing architecture. In our solution, we envisage to use the FIB of a Forwarding Engine as a *route cache* and to deploy a *centralized routing engine* that serves all the nodes of a sub-system. Fig. 4 reports an example of Lookup-and-Cache operations. Node *N* receives an Interest message for "ccn.com/text1.txt/chunk1". Since the FIB lacks the related route, the node temporarily queues the Interest message, lookups the route in a remote RIB, gets the routing information and stores it in the FIB, and then it can forward the Interest message. In what follows, we discuss the rationale underlying the Lookup-and-Cache architecture.

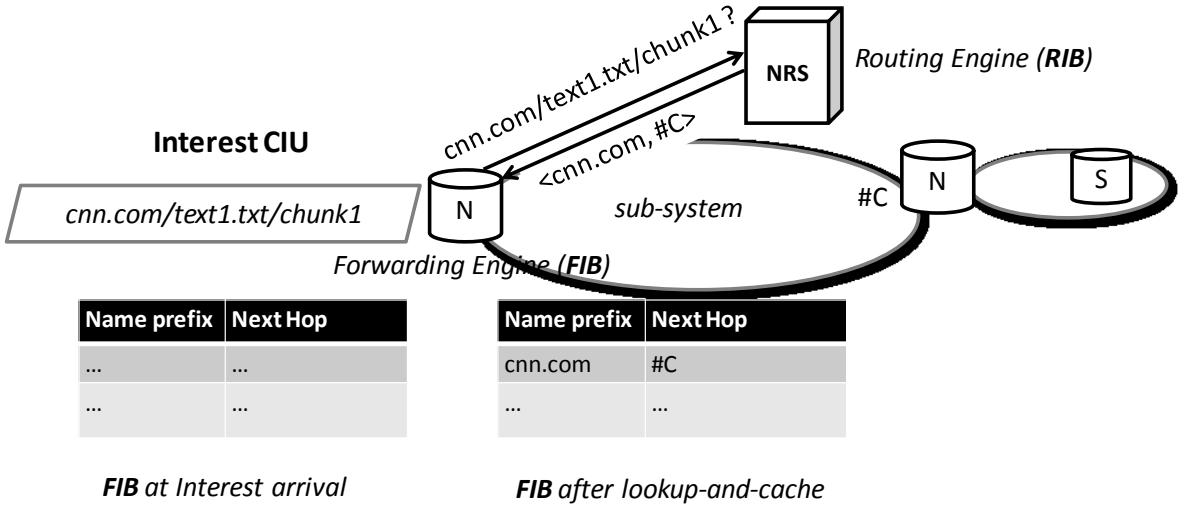

Fig. 4 – Lookup-and-Cache concept

FIB as a route cache – It is well-known that the relative frequency with which Web contents are requested follows the Zipf's law [35] and that there is a time and space locality of Web content interests. Therefore, a large number of *flows* of Interest messages that an ICN node should *concurrently* route-by-name refer to a small set of contents and, more important, these flows use an even much smaller set of ICN routes, since ICN routes address servers rather than single contents (see Section 3). In Section 6, we will show that the set of these *active-routes* can be comfortably stored in a SRAM memory. Therefore, we propose to use the FIB as a route cache, which should contain, at least, the entire set of active-routes[5]. When the FIB lacks a route, the node lookups the route in a "remote" RIB and then caches the route in its FIB. When all FIB rows are filled in, new routing entries may substitute old ones, according to a specific *route replacement algorithm*. Furthermore, a routing entry could be removed or updated by a *FIB-RIB consistency mechanism*.

Centralized Routing Engine - All ICN routes are contained in the RIB of a Routing Engine, which serves all nodes of a sub-system and runs on a centralized server, named *Name Routing System* (NRS) node. Thus, the cost of an expensive Routing Engine is taken for only one network device, rather than for all network nodes.

Since many Interest flows use a small set of active-routes, the temporal dynamics of active-routes is *slower* than the flow dynamics. Indeed, a route is used for a period of time that is greater or equal than/to the duration of a single flow. This limits the lookup rate that a centralized Routing Engine should deal with and, in Section 6, we show that this rate is easily supported by current technologies.

So far we have described the "data-plane" of our Lookup-and-Cache architecture, i.e. the procedures carried out to forward ICN messages. In addition to the data-plane, the Lookup-and-Cache architecture

[5] In addition, we could use pre-fetching algorithms to fill in unused FIB entries with most popular ICN routes. Pre-fetching aims at reducing the delay suffered by Interest messages for lookup procedures. ICN routes to be pre-fetched could be chosen based on a statistical analysis on lookup requests measured at the NRS node.

(as the IP one) needs "routing-plane" procedures that run on NRS nodes and whose goal is to setup the RIBs. The routing-plane is out of the scope of this paper; anyhow we point out that the our architecture does not impose a specific routing-protocol. For instance, we can support both name-based version of BGP, as suggested in [3], or DONA [2], where the DONA Resolution Handler (RH) has the same function of the NRS node.

We conclude the section by observing that, as it occurs in the current Internet for BGP messages, the ICN nodes should give highest priority to routing signalling (e.g., lookup and routing messages), to limit the number of failed communication attempts and the delay

5.1 Route Replacement Algorithm

When a node receives an Interest message for a given content and it is not possible to find a matching route in the FIB, we have a *route-cache-miss* event. In this case the node "may" lookup for the route in the remote RIB and store it in the FIB. Then, the node can forward the Interest message and the following ones that will use the same route. In case of a route-cache-miss event, the forwarding of one or more Interest messages will be subject to a route-lookup delay. Obviously, if the node does not perform a route lookup in the RIB, then the Interest message would be dropped and, eventually, retransmitted by transport level mechanisms.

When the FIB is not full, a node always performs a lookup in case of a route-cache-miss event and the new route is added to the FIB. When the FIB is full, the insertion of a new route implies the replacement of an old route. In this condition, the aim of the route replacement algorithm is to decide whether to ask the NRS node for the new route to be inserted and, if so, which old route to replace.

Since we have a network decoupling between the RIB and the FIB, an inefficient design of the route replacement algorithm would result in an excessive rate of route lookups, with a consequent worsening of delay performance (as more Interest messages will be subject to the route-lookup delay) and an increase of the load of the NRS node. To mitigate these issues, it would be desirable to replace *inactive*

routes, i.e. routes not used by the traffic that the node is forwarding at a given time. Consequently, the design of the route replacement algorithm should aim at solving two central problems: first, how to understand that a route is inactive and, second, how to behave in case of *FIB overload*, i.e. when there are no inactive routes and a new route needs to be added in the FIB to forward an incoming Interest message.

In the following, we: i) propose a novel simple route replacement algorithm, based on the estimation of an Inactivity Time Out (ITO); ii) analyze the Least Recently Used (LRU) replacement strategy, to understand how LRU deals with these two problems; iii) study the relationship between the route replacement strategy and the fulfilment of Quality of Service requirements.

We evaluate the performances of these two strategies in Section 7.4.

5.1.1 Inactivity Time Out (ITO) - route replacement algorithm

The ITO algorithm assumes that each route contained in the FIB has an inactivity time out (ITO), after which the route is considered inactive. At the current time `now`, if the time elapsed between `now` and the arrival time of the last Interest message of the route is greater than the inactivity timeout, then the route is considered inactive. The timeout value is evaluated by using the same algorithm of the TCP retransmission time out [25], where, instead of using round-trip-time measurements as input, we use measurements of inter-arrival times between two consecutive Interest messages of the same route. Details are reported in Appendix 1 of [47].

In case of FIB overload, the ITO algorithm is *partly non-preemptive*. An active-route cannot be removed from the FIB, unless the FIB is in a "pressuring" state since a period of time greater than a Maximum Pressuring Time (MPT).

The pressuring state starts when the entering of a new route in the FIB is prevented and ends when a new route is inserted in the FIB. We introduced the MPT parameter to avoid that active routes contained in the FIB block new traffic flows for an excessive time.

The ITO algorithm works as follows. When a new route needs to be inserted there are three possibilities: i) if there are inactive-routes (*FIB underload*), the least recently used inactive route is replaced by the new route; ii) if there are not inactive-routes (*FIB overload*) *and* the pressuring period is lower than MPT, the new route is not inserted and the incoming Interest message is discarded; iii) if there are not inactive-routes (*FIB overload*) and the pressuring period is greater than MPT, the least recently used active-route is replaced by the new route.

The ITO algorithm may temporarily prevents the forwarding of traffic. However, in case of route overload, the non-preemptive characteristic of the algorithm strongly limits the in/out flapping of routes in the FIB. As we will show in the next section, in/out flapping is harmful since it overloads the NRS node, increasing the time required by users to download contents and hampering long downloads.

5.1.2 Least Recently Used (LRU) - route replacement algorithm

With LRU, a new route that needs to be inserted in the FIB always replaces the least recently used one. LRU does not take into account if a route is active or not. Thus, in case of FIB overload, LRU is *pre-emptive,* as the least recently used route is replaced even if it is active.

5.1.3 Quality of Service

Quality of Service (QoS) refers to techniques used to provide specific performance levels to traffic classes with possible different service requirements [e.g., RFC 2990]. Examples include scheduling algorithms or access control policies devised to assure pre-defined loss and/or delay performance. Typically QoS is concerned with allocating transmission or storage resources. In our case, we have the rather unusual problem of controlling how different traffic classes can use FIB space. The limited caching space impacts both delay and jitter performance of traffic flows. When a new traffic flow reaches a node that does not have the corresponding ICN route in the FIB, this traffic suffers an *entry* delay composed of two components:

1) the time needed to store the route in the FIB, which depends on the replacement algorithm: it is equal to zero in case of LRU, while it depends on the state of activity of FIB entries and on the MPT timeout in case of ITO;
2) the time needed to carry out the lookup and cache procedure, which depends on the query processing time and on the network delay between FIB and RIB.

A traffic flow experiences the entry delay not only the first time that it reaches a node, but also each time that its route flaps in/out from the FIB, further worsening the jitter performance.

This means that a QoS algorithm should limit both the value of the entry delay and of the frequency of in/out flapping. As a simple example of such an algorithm we propose a modification of the ITO algorithm, named priority-ITO, whose performance will be discussed in section 7.4.

With Priority-ITO, in case of FIB overload, routes belonging to high priority traffic classes pre-empt routes belonging to low priority traffic classes. In this way, traffic of a class with priority #*K* sees the FIB space as it were used only by routes of classes with priority greater or equal to #*K*.

5.2 RIB-FIB consistency mechanism

Current IP routers implement a RIB-FIB consistency checker, whose logic does not require a careful optimization since RIB and FIB run in the same motherboard. In our case, FIB and RIB are physically separate. Thus, we need a consistency mechanism ensuring that the information cached in the FIB is consisted with the one in the RIB; the mechanism must also make an efficient use of the path connecting RIB and FIB.

The problem is well-known in the area of Web caching [45] and there are several mechanisms currently in use. Routing information is critical, so we need a *strong* level of consistency, i.e. the FIB must not contain stale entries. This implies that consistency mechanism based only on an expiration-time are not suitable. Polling mechanisms can not be used, as an ICN node can not afford to check the validity of FIB entries packet-by-packet. Consequently, we propose to use an *invalidation* mechanism, where the NRS

node keeps track of ICN routes cached in "its" FIBs: each time an entry of the RIB changes, the NRS notifies the related FIBs that the cached entry is no valid anymore. The NRS keeps track of the routes cached by a FIB thanks to lookup requests coming from the FIB. Each FIB entry has a long expiration time (ET), e.g. 10 min, after which the entry is removed. The routes cached in the FIB are the ones looked up in the last ET seconds. If their number is greater than the FIB size (*S*), then cached routes are the latest *S* looked up routes. The expiration time reduces the frequency of invalidation notifications since it limits the number of unpopular/occasional routes in the FIB. These routes generate most of the updates [46], as they are typically related to servers having unreliable (or poorly managed) connections to the rest of the Internet.

6. Feasibility check

In this section we show the feasibility of our architecture by using technology already available today and in the "fetching Web contents" scenario[6]. To this end we verify: i) that the capacity provided by current FIB technology is enough to store the expected number of active-routes; ii) that the route lookup rate can be supported by current database technology; iii) that the invalidation rate is compatible with processing technology and does not generate too much traffic over the RIB-FIB path.

On a given node and at a given time, an ICN route is "active" if there is at least one flow of Interest messages using that route. This concept is sketched in Fig. 5, where there are 3 flows of Interest messages toward "cnn.com". The route toward "cnn.com" becomes active at the start of the first flow and becomes inactive at the end of the last flow. In Fig. 5 there is also a single flow of Interests for "bbc.com", thus the related route activity has the same duration of the flow.

In the current Internet, a client sends TCP ACK and receives TCP segments from the Web server. In an ICN, a client sends Interest messages and receives Data messages from the ICN server, or from an en-

[6] To make this check we use real traces. Of course real traces come from the current Web. Thus, we can verify the feasibility for an ICN scenario with a number of name-prefixes in the order of current domain names, that is 10^8. Our initial target was 10^9, but obtained results show that current technology can support our solution with a wide margin (allowing 10^2 - 10^3 times increase), thus proving feasibility also for the case of 10^9 name-prefixes.

route cache. So, if a client used the ICN to download Web contents, then the traditional flows of TCP ACK messages would be replaced by a flow of Interest messages. Furthermore, on the base of our hierarchical naming assumption (see Section 3), the couple < IP destination address, destination Port> contained in TCP ACK messages would be replaced, in Interest messages, by a chunk-name that contains the name-prefix advertised by the Web server. For instance, assume that in the current Internet a host sends an HTTP request towards the domain name "cnn.com". The domain name "cnn.com" will be translated by DNS into an IP address, e.g. 157.166.226.25, a request will be sent to this address and then the data will be directed from 157.166.226.25:80 towards the requesting host, while a flow of TCP ACKs will be directed by the client to 157.166.226.25:80. In the proposed ICN scenario the flow of TCP ACKs would be replaced by a flow of Interest messages for chunks, whose names contain the "cnn.com" name-prefix.

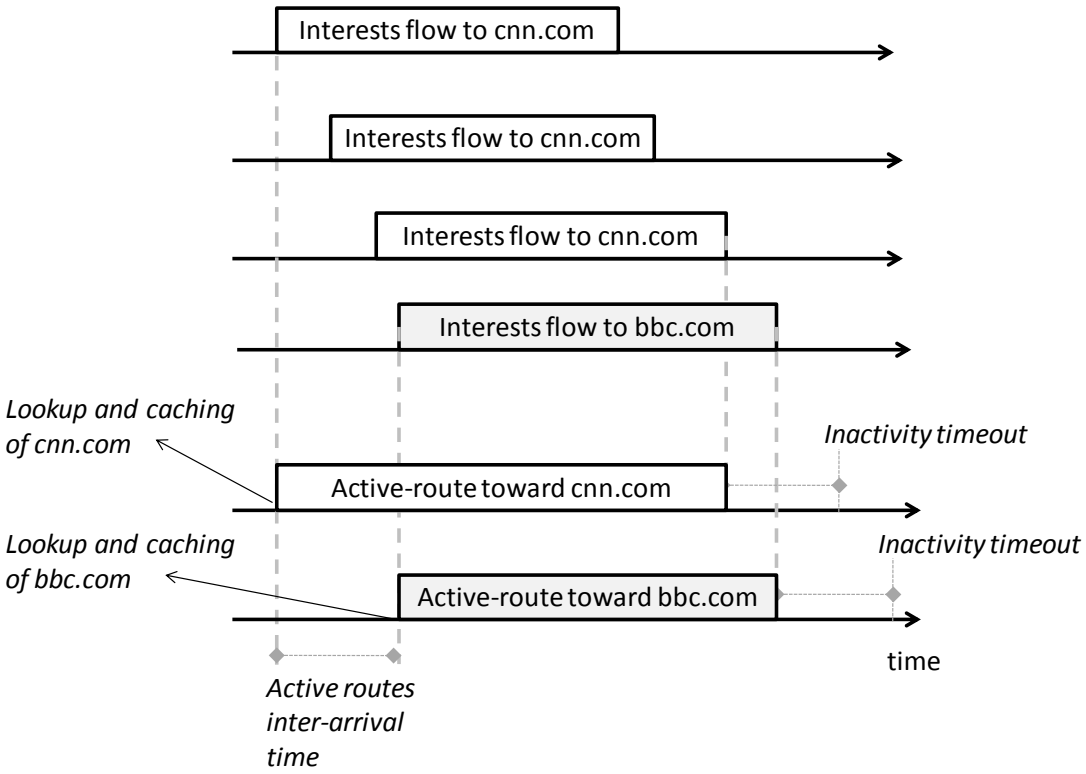

Fig. 5 –Flows and active-routes

Assuming a *one-to-one* relationship between an IP address and a domain-name[7], we can remap a trace of ACKs captured from current Internet in an equivalent trace of ICN Interest messages. In this section, we apply this *remapping* to real Internet traces, to estimate the number of active-routes of an hypothetical ICN, as follows.

We extract from the trace TCP ACKs with destination port equal to 80. We consider each ACK as if it were an Interest message for a chunk available at the destination Web server, with a one-to-one correspondence between ACK and Interest[8]. Following our naming scheme (Section 2), the first Component of the chunk-name is the domain name of the Web server[9]. We do not need other Components of the chunk-name, as routing-by-name operations consider only the first Component of the chunk-name (e.g., "cnn.com")[10].

The remapped trace of Interest messages feeds a simple algorithm, which infers the route activity by using an inactivity timeout of 1s. The start time of a route activity is the receiving time of the first Interest message; the route is considered inactive if there are no related Interest messages for more than 1 second; the end-time of a route activity is the receiving time of its last Interest message.

Tab. 1 summarizes our results. The Equinix-sanjose-* and Equinix-chicago-* traces are captured on a 10 GigE interfaces of a tier-1 ISP. The Mawi-* traces are captured on a trans-Pacific line operating at 150 Mbit/sec. The Rome-Tor-Vergata trace is captured on the 1 GigE interface of the router gateway of our University, which is a tier-3 network.

---

[7] We observe that the assumption of a *one-to-one* relationship between an IP address and a domain-name may lead to an underestimation of the number of active-routes and of the lookup rate if there are Web hosting services, where the same IP address may serve several domain-names. In Appendix II of [47], we present a simulation model that takes into account Web Hosting services. Simulation results show that Web hosting services increase the number of active-routes and the lookup rate with respect to the values computed with the *one-to-one* assumption; nevertheless the order of magnitude remains the same, hence the conclusion on the feasibility remains valid.

[8] Interest messages refer to chunks, which are typically greater than TCP segments. The number of Interest messages would be lower than the number of ACKs. However, the one-to-one correspondence between ACK and Interest allows tracking the activity of flows and routes, which is exactly what we need.

[9] We have only anonymized traces, from which we can not derive the actual domain name of the server; so we used a fake domain name equal to the anonymized IP address of the server, considered as a flat string.

[10] If we needed to know the full chunk-names, we should have performed an HTTP level analysis of the traces, rather than limiting ourselves to the analysis at TCP level.

As expected, the interfaces of tier-1 ISP have the highest number of active-routes, whereas the interface at our University the lowest one. Even in the worst case of the Equinix-sanjose-dirA trace, the maximum number of active-routes is in the order of 10^3 and, as discussed in section 3, this value is much lower (by a factor of 10^3) than the capacity provided by an off-the-shelf SRAM based FIB, i.e. 10^6 ICN routing entries.

Trace id	Date (m-g-y, h:m)	Trace duration	Average n. of active-routes	Max n. of active-routes	Average active-routes inter-arrival	Ref.
Equinix-sanjose-dirA	11-18-2010, 13:50	60 s	2582	2681	1.2 ms	[12]
Equinix-sanjose-dirB	11-18-2010, 13:50	60 s	1293	1353	2.8 ms	[12]
Equinix-chicago-dirB	10-29-2010, 13:06	60 s	1197	1283	1.7 ms	[12]
Mawi-1	11-28-2011, 14:00	15 min	236	289	5.9 ms	[13]
Mawi-2	11-29-2011, 1400	15 min	246	303	4.1 ms	[13]
Rome-Tor-Vergata	03-04-2011, 15:06	9 min	176	227	7.3 ms	[37]

Tab. 1 – Internet trace analysis report

Fig. 6 shows the number of active-routes versus time for the Equinix-sanjose-dirA trace. The number of active-routes has limited variation around its average value of 2582. This simplifies the dimensioning of the FIB size, which can be set close to the observed mean, without requiring a large margin [38].

The small number of active routes is the result of the Zipf nature of the Web [35]: a wide set of flows refers to a limited set of "popular" contents and uses an even more smaller set of ICN routes. In fact, a flow-level analysis[11] reveals that the whole trace contains about 2.6 millions of Interest flows, i.e. of content downloads. These 2.6 millions of Interest flows use only 11000 routes[12]. Fig. 8 shows the cumulative density function (CdF) of the *route* popularity, versus the route ranking (*k*). The value of

[11] We measured the number of active-flows using the same methodology used for the number of active-routes, i.e. based on an inactivity time out of 1 sec. An active-flow is identified with the triple <client-address, server-address, client port>.

[12] The average number of flows active at a given time *t*, i.e. the number of *active-flows*, is equal to 27865 versus a number of active-routes equal to 2582.

CdF(k) is the probability that a flow uses a route that belongs to the first k mostly used routes. We also report the CdF of the Zipf distribution, which is known to be representative of *content* popularity. The figure shows that most flows use a small set of popular routes; route popularity is even more skewed than Zipf, since an ICN route addresses servers rather than single contents, and different popular contents may be offered by the same server (e.g., YouTube). Roughly one tenth of the entire set of 11000 routes accounts for about 96% of flows (this is consistent with previous work on IP figures [38]).

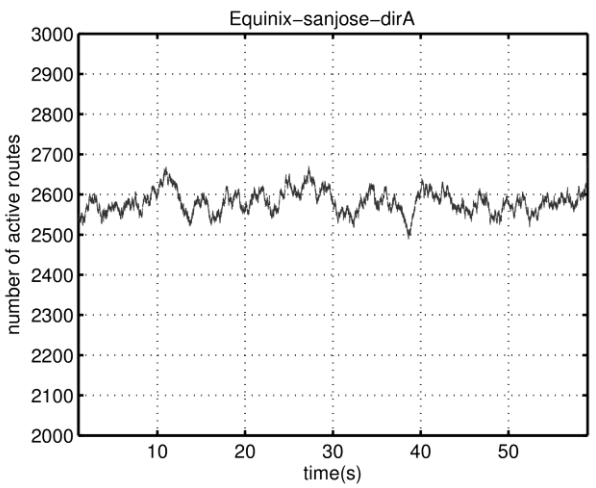

Fig. 6 – Number of active-routes for the Equinix-sanjose-dirA trace

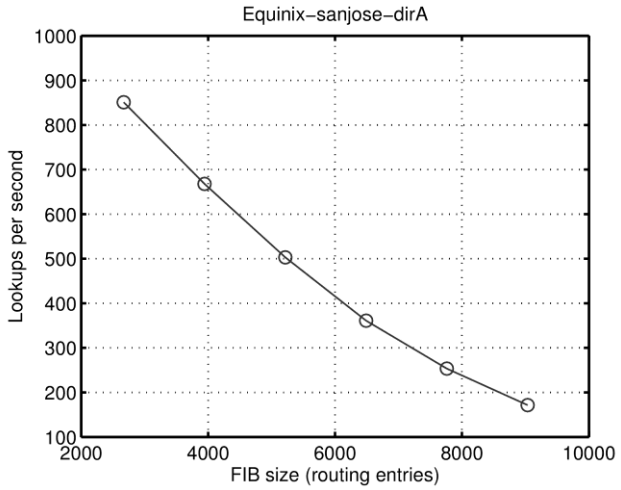

Fig. 7 – Lookup rate versus FIB size for the Equinix-sanjose-dirA trace

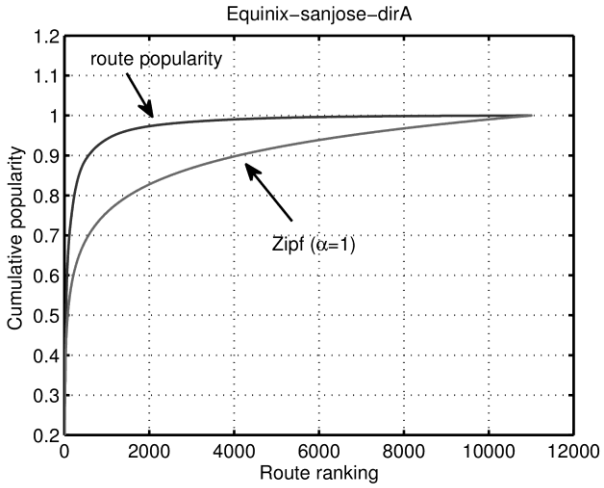

Fig. 8 – Cumulative route popularity for the Equinix-sanjose-dirA trace

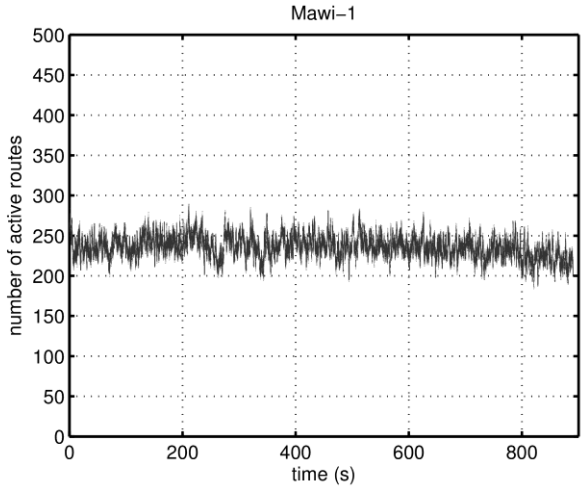

Fig. 9 – Number of active-routes for the Mawi-1 trace

Let us now investigate if current database technology can support the required lookup rate. Tab. 1 reports that the average inter-arrival time between the starts of two consecutive active-routes is in the order of milliseconds, for the considered traces. When the FIB memory is dimensioned for containing all active-routes, the inverse of the active-routes inter-arrival time is an upper bound of the lookup rate. Indeed, we need a lookup at the start of the route activity only if that route is not already cached in the FIB. Therefore, an average active-route inter-arrival time in the order of few ms implies a lookup rate in the order of 1000 lookups per second, in the worst case. This value is easily achievable with current database technology. For instance, we have implemented an NRS node with a Bind9 server, running on an old Linux laptop with an Intel Pentium Processor M at 1.4 Ghz and we measured a sustainable rate of about 15 000 lookups per second.

We investigated also the effectiveness of FIB *over-provisioning*, to reduce the lookup rate. A FIB is said to be over-provisioned, when it has a capacity greater than then maximum number of expected active-routes. For this analysis we used an ideal route replacement policy that randomly replaces inactive-routes. Fig. 7 shows the resulting lookup rate vs. the FIB size for the Equinix-sanjose-dirA trace. The lookup rate decreases as the FIB size increases because each route frequently switches on and off. The trace contains a number of route activations/deactivations equal to about 54000, whereas the number of unique routes is about 11000. This temporal correlation among route activities implies that the increase of the FIB size has a relevant impact on the cache hit-ratio and, hence, on the lookup rate.

We also note that the improvement of the hit-ratio with the cache size is larger in our scenario of a cache of routes with respect to a cache of contents. In the latter case the hit-ratio grows only in a log-like way versus the cache size [35].

Fig. 9, Fig. 10 and Fig. 11 show the same performance discussed above for the Mawi-1 trace. This trace, and all other analyzed traces (not reported here), lead to the same conclusions just reported.

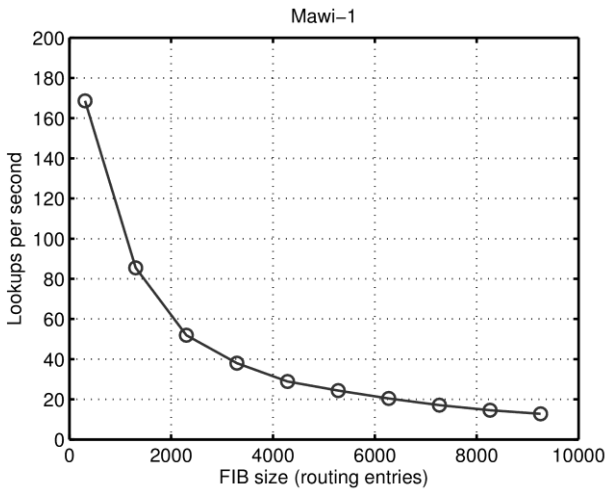

Fig. 10 – Lookup rate versus FIB size for the Mawi-1 trace

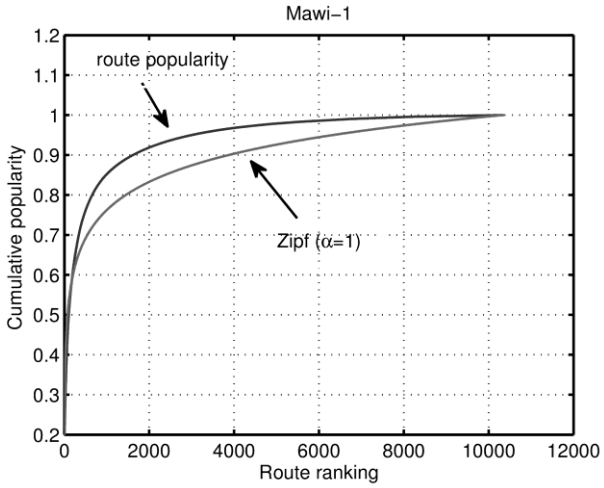

Fig. 11 - Cumulative route popularity for the Mawi-1 trace

Let us now investigate the feasibility of the invalidation approach described in Section 5.2. We start by measuring what occurs today in the IP routing-plane and then we exploit to derive conclusions valid for an ICN.

BGP statistics reported in [30] show that the sum of the average number of IP prefix updates and withdraws per second is quite close to 3. This rate is an upper bound of the rate at which an IP FIB may change. In fact, only a subset of BGP updates causes a modification of the IP FIB, depending on the topological location of the router. For instance, the FIB update rate of the router AS65000 [30] is about one per second.

As of today, the number of Internet domain-names is about $2*10^8$ and the number of IP prefixes is in the order of 400k. Assuming a uniform distribution of domain-names on IP prefixes, an IP prefix serves, on average, 500 domain-names.

In ICN the routing-plane deals with name-prefixes that, in our scenario, are domain-names. Therefore, the number of name-prefix updates and withdraws handled per second by the ICN routing-plane would be in the order of 1500 (i.e., 3*500).

A name-prefix update triggers an invalidation procedure only if the name-prefix is contained in the FIB. Therefore, the invalidation rate is equal to the name-prefix update rate multiplied by the probability that

a name-prefix is contained in the FIB. This probability is roughly equal to the ratio between the FIB size and the number of domain-names, i.e. $2*10^8$. In the deployment scenario we are considering, the FIB size in the order of 10^6 entries; thus the invalidation rate is in the order of 7.5 invalidations per second (i.e. $1500 * 10^6 / (2 * 10^8)$); this rate is easily supported by current technologies.

7. Testbed analysis

We deployed a test-bed to analyze performance and limits of our architecture.

7.1 Hardware setup

Fig. 12 shows the network architecture used for the tests. We have two sub-systems and each sub-system is an IP network connected by a 100 Mbit/s Ethernet switch. The sub-system *A* contains three ICN clients. The sub-system *B* contains an ICN server and an NRS node. The sub-systems are interconnected by node *N*, equipped with two network interfaces.

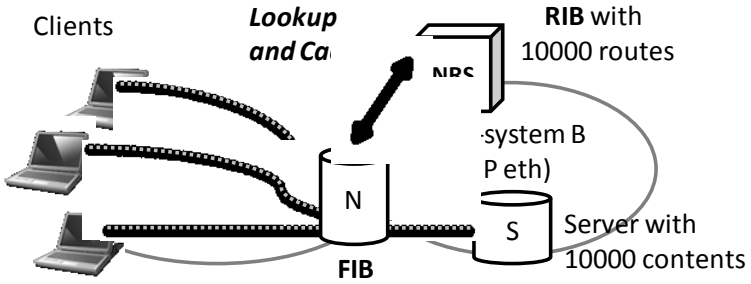

Fig. 12 –Testbed setup

7.2 Software setup

All devices run Linux OS. Clients, node *N* and server run a modified version of the CCNx software release 0.5.0 [17], where we have implemented our Lookup-and-Cache routing, with both ITO and LRU route replacement strategies.

The FIBs of clients have a single default route toward node *N*. Hence, clients do not perform Lookup-and-Cache procedures. Node N does not have a default route and uses the Lookup-and-Cache routing-by-name mechanism to feed its FIB, up to a fixed size of 100 routing entries.

The server has a repository containing 10 000 contents; the content size follows a Pareto distribution (k=133, $\alpha = 1.1$) [33] and each content is divided in chunks of 4 kBytes [13].

The NRS node has a RIB indexing all 10 000 contents and the next-hop value of each RIB entry is the IP address of the server. The RIB has been implemented by means of the Bind-9 DNS server; therefore, the exchange of Lookup-and-Cache messages complies with the DNS protocol. To emulate the delay of a wide area network, we artificially set up a two-ways delay of 100 ms between node *N* and the NRS node. In addition to this network delay we measured, a-posteriori, an average delay of 60 ms, spent to accomplish the set of *local* lookup-and-cache procedures, such as inserting the route in the FIB, de-queuing waiting Interest messages, etc.. Summing up, a lookup-and-cache operation adds an average delay of 160 ms to an incoming Interest message that does not find the route in the FIB.

In all nodes we set the size of the "content" cache to zero, so there is no influence of *content* caching on the evaluated performance. Moreover, we configure the CCNx InterestLifeTime parameter at 1 sec.

7.3 Persistent workload

The number of active-routes is a crucial variable to highlight performance limits of the Lookup-and-Cache routing, as well as to compare different route replacement strategies. Therefore we set up a *persistent* workload model that generates a constant number of active-routes. The workload keeps fixed the number of concurrent downloads performed by clients. Each download fetches a content never downloaded before, so the number of downloads is equal to the number of active-routes. Each time a download ends, after 50 ms a new download starts. If the first Interest message is dropped, the message is periodically resent each 2 seconds. To download a content clients use the ccncatchunks2 utility [17], configured with a maximum congestion window equal to 4 chunks. For instance, a scenario of 30 active-routes is realized by concurrently running 10 ccncatchunks2 instances on three clients, with each

[13] We have also analyzed cases of contents having all the same size and considering different sizes (500 kB, 1MB, 10MB); the resulting behavior is pretty much the same.

ccncatchunks2 instance downloading a different content. This goes on for all the test (10000 downloads).

7.4 Testbed results

We measured Lookup-and-Cache performance with a FIB limited to 100 entries, by using both ITO and LRU. As a benchmark, we compare these performance with the one obtained for an *unlimited* FIB, where *all* routes are properly preloaded. Each test is repeated 5 times and we measured both average performance and 95% confidence intervals.

7.4.1 Analysis of delays and lookup rates

Fig. 13 reports the average download time versus the number of active-routes. Fig. 14 reports the average number of lookups per download, equal to the ratio between number of lookups performed by node *N* and overall number of downloads. We first analyze an unloaded FIB (number of active-routes lower than 100) and then an overloaded FIB.

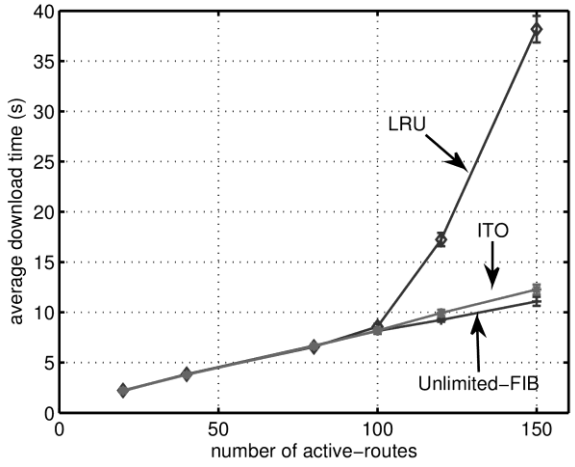

Fig. 13 – Average download time versus number of active-routes

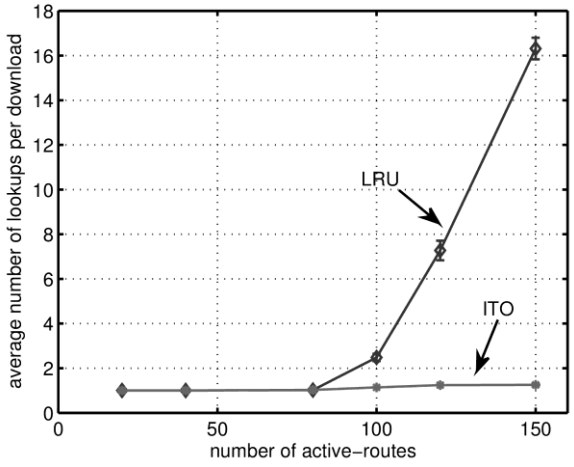

Fig. 14 – Lookup rate versus number of active-routes

7.4.1.1 Unloaded FIB

As expected, Lookup-and-Cache works well when the number of active-routes is lower than the FIB size (100 routes). Download times are comparable with the ones measured in the unlimited-FIB case and the number of lookups per download is close to one. This means that after a first lookup-and-cache cycle, a

route is held in the FIB during the download time. The download time is the sum of the initial lookup delay (160 ms, if the route is not already in the cache) plus the content download time. The latter time depends on the available link capacity and therefore it linearly increases with the number of competing downloads (active routes). In these unloaded conditions ITO and LRU perform almost in the same way.

7.4.1.2 Overloaded FIB

Performance start to decrease when the number of active-routes is equal or to greater than the FIB size. The more the overloading, the greater the performance degradation, with respect to the unlimited-FIB case. In these conditions, the difference between the route replacement algorithms clearly shows up.

With ITO, the download delay is a bit greater than in the unlimited-FIB case and the number of lookups per download remains quite limited. These results are the consequence of the non-preemptive behaviour of the ITO algorithm. With a FIB of 100 entries, only 100 downloads actually transfer data, while other downloads are waiting for their route to be cached in the FIB. The caching of a route will be possible when one of the 100 downloads ends (or the MPT timeout of 1 sec expires, but this occurred very rarely in our experiment). Thus, ITO schedules the use of transmission resource for 100 downloads at a time. Differently, in case of unlimited-FIB, all downloads transfer data concurrently. Therefore, the two cases mainly differ on *how* transmission resources are scheduled for downloads and this has a minor impact on average delay performances.

Fig. 14 shows that ITO limits the number of lookups per download to 1.2, also in overloaded conditions. The rise of the lookup rate is due to transport level time outs, during which a route timeout set by ITO may wrongly elapse. In these cases, the route is removed from the FIB, and then briefly reinserted, which increases the number of lookup per content.

With LRU, the number of lookups per download and the download times significantly increase. When the number of active routes is equal or to greater than the FIB size, the LRU algorithm is *pre-emptive* as it replaces FIB entries associated to active routes. This yields an in/out flapping of routes in the FIB. The

flapping suddenly increases download delay and number of lookups per content. Albeit not tested, we argue that if we further increased the overload, the number of lookups per contents would approach the upper bound of one lookup per Interest message. The upper bound is reached in the limit case in which a route is hold in the FIB just for forwarding a single Interest message.

7.4.2 Fairness analysis

In this section we analyze the fairness performance of our architecture. We evaluate the download bit-rate (goodput) as a function of the content size, to check if large contents (e.g. 10 Mbytes) are downloaded with a different goodput with respect to small contents (e.g. 200 kbytes).

We organize contents in different *classes*, where each class includes contents whose size is within a given range, (e.g., 0-500 kbytes). For each class, we evaluate the average goodput achieved by downloading contents of that class. Then, we compute the Jain's fairness index of the class goodputs, defined [34] as follows:

$$F_J = \frac{(\sum_{i=1}^{N} g_i)^2}{N \sum_{i=1}^{N} {g_i}^2}$$

where g_i is the average goodput obtained by downloading contents of the i-th class and N is the number of classes. Perfect fairness is achieved when $F_J = 1$ and absolute unfairness is achieved when $F_J = 1/N$.

We first discuss the case of unloaded FIB and then the overloaded FIB.

7.4.2.1 Unloaded FIB

Fig. 15 reports the Jain's fairness index versus the number of active-routes, computed by arranging contents in classes with a range of 500 kbytes, and for the two route replacement strategies. The first class includes contents with a size in the range {0, 500 kbytes}, the second class is includes contents with a size in the range {500 kbyes, 1Mbytes}, and so forth. For a number of active-routes lower than the FIB size (i.e. 100 routes), ITO and LRU have similar performance, which are quite close to the

unlimited-FIB case. Therefore we conclude that, in unloaded conditions Lookup-and-Cache provides good fairness performances.

The fairness index improves by increasing the number of active–routes from 20 to 40. To explain this behaviour we use Fig. 16 and Fig. 17, where we report the goodput perceived by three classes of contents {0, 500kbytes}, {500 kbytes, 2Mbytes}, {2Mbytes, 10Mbytes} with 20 and 40 active routes, respectively.

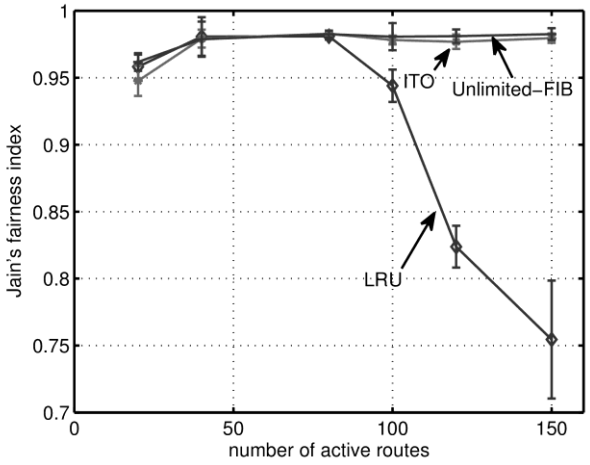

Fig. 15 – Jain's fairness index versus number of active-routes

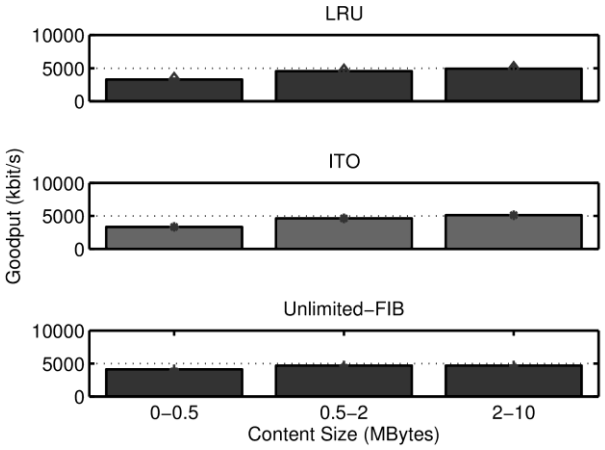

Fig. 16 – Goodput versus content size in case of 20 active-routes

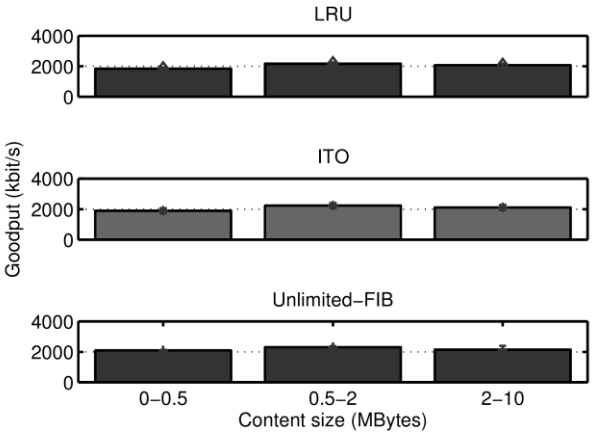

Fig. 17 – Goodput versus content size in case of 40 active-routes

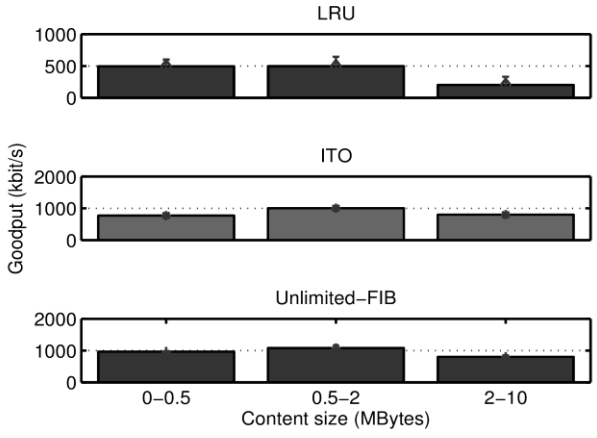

Fig. 18 – Goodput versus content size in case of 150 active-routes

With 20 active-routes, the download of small contents is slightly penalized and we argue this is due to processing delays. Indeed, download rates are high and the processing delay has a significant impact on

the goodput achieved by downloading short contents. With respect to the unlimited-FIB case, the unfairness behaviour is worsened by the Lookup-and-Cache routing, since each download suffers an additional processing delay, due to the execution of lookup-and-cache procedures. This suggests that network paths connecting the NRS node with other network nodes should be properly dimensioned in order to limit as more as possible the round trip time.

By increasing the number of active-routes, the download rates decrease as well. Consequently, processing delays have a lower impact on the goodput of small contents and the fairness level increases. This can be seen from Fig. 17, where we have a number of active-routes equal to 40 and we observe a higher fairness.

7.4.2.2 Overloaded FIB

When the number of active-routes is equal to or greater than the FIB size, we observe that ITO obtains a fairness performance very close to that of the unlimited-FIB case. Conversely, the fairness performance of LRU significantly decreases by increasing the number of active-routes. This decrease is due to the in/out route flapping events that penalize the download of long contents. Fig. 18 reports the goodput perceived by different classes of contents in the case of 150 active-routes. With LRU, the greater the content size, the lower the goodput.

Finally, we report a specific measurement showing the effectiveness of the ITO MPT timeout to avoid traffic starvation in presence of routes that remain active for a long time. We consider a FIB size of 3 entries, and 4 downloads of different *unlimited* contents, which start at time 0, 0.2, 0.4, 0.6 s, respectively. The ITO algorithm has an MPT equal to 1 second. Fig. 19 reports the presence (in) and the absence (out) of routes in the FIB versus time. We observe that when the route of a download is out from the FIB, this condition persists for about 1 second, i.e. the value of MPT timeout.

7.4.3 QoS analysis

In this section, we evaluate the priority-ITO algorithm described in Section 5.1.3. We consider three classes of traffic and setup the following test: with reference to the scenario in Fig. 12, the client n.1 generates traffic of class 1, client n.2 of class 2 and client n.3 of class 3. At the start of the test, all clients request 50 different contents, whose length is fixed to 10 Mbytes. The FIB size is 50, so only 50 downloads can be concurrently supported. Fig. 20 reports the average download time measured for the different classes. As expected, the greater the class priority, the lower the delay.

During the test we observed three phases. In a first phase, the 50 downloads of class 3 fill the FIB; thus, there are only downloads of class 3, while downloads of other classes are waiting for a FIB space. In a second phase, downloads of class 3 end and the 50 downloads of class 2 fill the FIB. Thus, there are only downloads of class 2, while downloads of class 1 are waiting again for a FIB space. In a third phase, also downloads of class 2 end and downloads of class 1 can start.

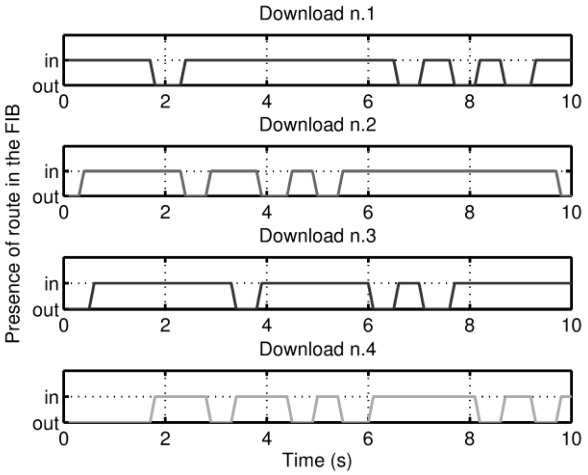

Fig. 19 – Presence and absence of routes in the FIB

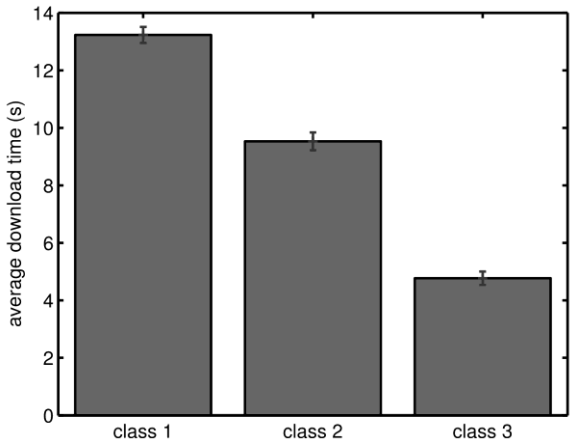

Fig. 20 – Goodput versus class of traffic, in case of priority-ITO

8. Support of interactive/conversational services

In this paper we focused on ICN as a solution for content retrieval, and specifically for fetching Web contents. This scenario is perfectly tailored for an ICN; indeed, an ICN natively supports a *data pull* model, where data transfer is controlled by receivers that pull information from sources.

This leaves open the question about interactive/conversational services, which instead best fit the *data push* model, where data transfer is controlled by sources that push information to receivers. Shall we support also interactive/conversational with ICN or continue using IP (or a parallel architecture) for them?

The question is tied to the possible deployment alternatives for ICN. In line of principle, an ICN could be deployed either by an *overlay approach* (ICN on top of the IP layer) or by a *clean slate approach* (ICN on top of layer-2 technologies, removing IP). In [22] we proposed also a third approach, named *integration approach* (ICN functionality integrated in the IP layer by means of a novel IPv4 option or by means of an IPv6 extension header). In this integration approach, ICN functionality is added to IP in a backward compatible way, so that most advantages of ICN can be exploited without dropping IP. In this scenario, *content-oriented* communication patterns can be supported by the proposed ICN extensions to IP protocols, while *interactive/conversational* communication patterns can continue to run over traditional IP. The same is true for ICN deployed as an overlay approach, on top of IP. In that case, since IP would remain underneath, *interactive/conversational* services could still exploit IP.

Instead, if ICN is deployed by following a clean slate approach, then of course it is necessary to envisage suitable mechanisms to support such services. For instance, [48] provides a way to support voice over ICN. Another example is suggested by ourselves in [8] and [22], where our CONET supports services based on data-push model, like the interactive/conversational ones, by means of so called *named-saps*. A named-service-access-point (i.e. named-sap) is a network endpoint addressed by a name, through which an upper layer entity (e.g. a server or a client) sends and receives data. In the current Internet, for instance, TCP port n. 80 is the default service-access-point for HTTP servers.

In the case of a named-sap (e.g. the logical port of a server) CONET provides the means to exchange data between a requesting upper layer entity and the upper layer entity addressed by such named-service-access-point. Thus, the CONET offers two mechanisms for the delivery of data:

i) When users need to access data (e.g., documents, files, etc.), the CONET allows them to download actual content. In this case the user asks the network to provide a resource with a given name and the network provides her with the actual content, without involving any other upper layer functionality.

ii) When it is necessary to support service sessions for upper layer entities (e.g. a client-server couple), the CONET couples the local upper layer entity with the named-service-access-point for the "best" remote upper layer entity providing the service and goes on to support an interactive exchange of data between these two upper layer entities.

9. Related work

The use of the FIB as a route cache and the centralization of the Routing Engine have been already investigated in the context of traditional IP networking. However, we did not find in the literature a proposal of an IP architecture that puts together these two concepts.

In the far 1988, Feldmeier [23] proposed adding a route-cache to a router. The cache was used to speed up the lookup operation of forwarding process. The router however had a local RIB, looked up in cases of route-cache miss. The route replacement algorithms were LRU and FIFO, and LRU achieved the best performance. Differently from [23], in case of our architecture, the RIB is provided by a remote device and, as we discussed in sections 5.1, such a network decoupling between the FIB and the RIB imposes a rethinking of route replacement algorithms.

In [38], the Authors revised route caching as an instrument to face the growing of IP routing tables with limited, but expensive, FIB memories. They proposed to use the FIB to cache "uni-class" (/24) IP prefixes and, as in [23], they assume to have a local RIB. An analysis based on real Internet traces revealed that the working set of /24 prefixes used in limited time windows, ranging from 1 min to 60 min, is abundantly lower than the whole set of CIDR address. Therefore the authors concluded that caching would be, not only *necessary*, but also *possible*. A performance comparison between the LRU

and Least Frequently Used (LFU) replacement policies showed that LRU provided better performances. In this IP scenario the FIB contains /24 routes towards both clients and servers. In our ICN scenario, the FIB contains only the routes toward the servers. So, the trace analysis of [23] deals with all /24 routes, whereas our analysis deals with /32 routes towards Web servers. For this reason, numerical results are different (we measured a lower number of active-routes) but consistent.

In [39], the authors proposed to separate IP routing from routers. The routers should forward packets, while a centralized Routing Engine should select routes on behalf of IP routers in each autonomous systems (AS) and exchange reachability information with other Routing Engines of other ASs. This approach was proposed to reduce the complexity of the distributed computation of the routes, since only one entity per AS participates to the routing plane. When a Routing Engine computes a new route, it *pushes* the route in the FIBs of the AS routers. Differently, in our ICN architecture, the FIBs *pull* the routes from the Routing Engine. The authors also pointed out that separating routing states from the routers can potentially introduce problems of robustness, speed, and consistency, which, however, are solvable by using current technology.

The approach of separating packet forwarding from control decisions is also at the basis of the SDN (Software Defined Networking) paradigm [40][41], which has been proposed quite recently and is now gaining momentum. Our Lookup-and-Cache architecture does not require SDN to be implemented. However, the node-to-NRS interface could be easily remapped on the OpenFlow [40] switch-to-controller API.

10. Conclusions

An ICN network requires a careful design of the routing-by-name architecture, since it needs to support a huge number of routes. This is due to the need of addressing contents, which are many, and have names that are difficult to aggregate in routing tables, since they do not include topology information. The great number of routes practically prevents to reuse the well-tested architecture of an IP router.

Current FIB technology is unable to host all routes, and the RIB would be extremely costly. Our architecture is a first answer to this challenge. It exploits the Zipf nature of the Web. It works with off-the-shelf technologies. It copes with the limitation of FIB capacity by using the FIB as a route cache, which stores a limited subset of routes, while all routes are stored in the RIB of a remote and centralized Routing Engine.

Whenever possible, the FIB size should be over-dimensioned with respect to the expected number of active-routes. In these conditions, well-known cache replacement strategies, such as LRU, provide performance similar to the one of unlimited-FIBs. An analysis based on real Internet traces revealed that the number of active-routes is very limited. Thus, current SRAM technology can provide FIBs of the necessary size, allowing also over-provisioning.

The LRU replacement policy, and other policies that remove active-routes from the FIB should be avoided, if over-dimensioning the FIB is not possible. In cases of FIB overload, LRU adversely impacts download delay, lookup rate and fairness, because of in/out flapping of the routes from the FIB. In these conditions, other replacement policies, such as ITO, which do not remove active-routes from the FIB perform better. ITO has performance close to the one of an ideal unlimited-FIB.

11. ACKNOWLEDGMENTS

This work was partly supported by the EU in the context of the FP7 CONVERGENCE project [8].

REFERENCES

[1] D. Cheriton, M. Gritter, “TRIAD: a scalable deployable NAT-based internet architecture”, Technical Report (2000)”

[2] T. Koponen, M. Chawla, B.G. Chun, et al.: “A data-oriented (and beyond) network architecture”, ACM SIGCOMM 2007

[3] V. Jacobson, D. K. Smetters, J. D. Thornton et al., ”Networking named content”, ACM CoNEXT 2009

[4] D. Smetters, V. Jacobson: “Securing Network Content”, PARC technical report, October 2009

[5] D. Trossen, M. Sarela, and K. Sollins: "Arguments for an information-centric internetworking architecture" SIGCOMM Computer Communication Review, vol. 40, pp. 26-33, 2010

[6] PURSUIT project website: www.fp7-pursuit.eu

[7] SAIL project website, http://www.sail-project.eu/

[8] CONVERGENCE website: www.ict-convergence.eu

[9] COMET project website: www.comet-project.org/

[10] Named-Data Networking (NDN) project website, http://named-data.org/

[11] COAST project website: http://www.coast-fp7.eu/

[12] CAIDA Internet Trace Storage, https://data.caida.org/datasets/passive-2010/

[13] MAWI Working Group Traffic Archive, http://mawi.wide.ad.jp/mawi/samplepoint-F/2011/

[14] X. Tang, S.T. Chanson, "Coordinated en-route web caching", IEEE Transactions on Computers, Volume 51 Issue 6, June 2002

[15] A. Kuzmanovic, E.W. Knightly. "Receiver-Centric Congestion Control with a Misbehaving Receiver: Vulnerabilities and End-point Solutions", Elsevier Comput. Network. 2007, 51, 2717–2737.

[16] D. Perino, M. Varvello, "A Reality Check for Content Centric Networking", ACM SIGCOMM 2011, Workshop on Information-Centric Networking

[17] CCNx project web site: www.ccnx.org

[18] D. Meyer , L. Zhang , K. Fall , "Report from the IAB Workshop on Routing and Addressing", IETF RFC 4984

[19] A. Ghodsi, T. Koponen, B. Raghavan, S. Shenker, A. Singla, and J. Wilcox , "Information-Centric Networking: Seeing the Forest for the Trees", in Proc. of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X), Cambridge, Massachusetts

[20] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, N. Blefari-Melazzi, "Transport-layer issues in Information Centric Networks", ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2012), August 17, 2012, Helsinki, Finland

[21] B. Baccala, "Data-oriented Networking" Internet draft, IETF, August 2002.

[22] A. Detti, N. Blefari-Melazzi, S. Salsano, M. Pomposini, "CONET: A Content Centric Inter-Networking Architecture", ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2011), August 19, 2011

[23] D.C. Feldmeier, "Improving gateway performance with a routing-table cache", in Proc. of IEEE INFOCOM 1988

[24] G. Trotter , "Terminology for Forwarding Information Base (FIB) based Router Performance", IETF RFC 3222

[25] V. Paxson, M. Allman, "Computing TCP's Retransmission Timer", IETF RFC 2988

[26] Xiaoliang Zhao, Member, IEEE, Dante J. Pacella, and Jason Schiller, "Routing Scalability: An Operator's View", IEEE Journal on Selected Areas in communications, vol. 28, no. 8, October 2010

[27] "Cisco Carrier Routing System", available at http://www.cisco.com/en/US/prod/collateral/routers/ps5763/prod_brochure0900aecd800f8118.pdf

[28] "Juniper T Series Core Routers" available at http://www.juniper.net/elqNow/elqRedir.htm?ref=http://www.juniper.net/us/en/local/pdf/datasheets/1000051-en.pdf

[29] "Verisign 8-k Current Report", available at https://investor.verisign.com/secfiling.cfm?filingID=1193125-10-213453

[30] "BGP Routing Table Analysis Report", available at http://bgp.potaroo.net

[31] "Daily estimated size of World Wide Web", http://www.worldwidewebsize.com/

[32] "BGP Routing Table Analysis - DIX-IE Data", http://thyme.apnic.net/current/

[33] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", In Proc. of ACM SIGMETRICS '98

[34] R. JAIN. The Art of Computer Systems Performance Analysis. John Wiley & Sons, 1991

[35] L. Breslau et al., "Web Caching and zipf-like Distribution: Evidence and Implications", in Proc. IEEE INFOCOM, 1999

[36] ACM SIGCOMM Workshop on Information-Centric Networking (ICN-2011), http://www.neclab.eu/icn-2011/program.html

[37] http://netgroup.uniroma2.it/Andrea_Detti/Lookup-and-Cache/

[38] Changhoon Kim, Matthew Caesar, Alexandre Gerber, and Jennifer Rexford, "Revisiting route caching: The world should be flat," in Proc. Passive and Active Measurement Conference, April 2009.

[39] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers", in Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture, August 2004.

[40] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner, “OpenFlow: Enabling Innovation in Campus Networks”, ACM SIGCOMM Computer Communication Review, Volume 38, Number 2, April 2008

[41] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martin Casado, Nick McKeown, and Scott Shenker, “NOX: Towards an Operating System for Networks”, ACM Computer Communications Review, April 2008

[42] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network", in Proceedings of ACM SIGCOMM, August 2001

[43] Alexa Web Informatin Company, “Top 1,000,000 Sites” available at http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

[44] J. Jung, E. Sit, H. Balakrishnan, R. Morris “DNS Performance and the Effectiveness of Caching”, IEEE/ACM Transactions on Networking, Vol. 10, No. 5, October 2002

[45] Gwertzman, J., Seltzer, M., World-Wide Web Cache Consistency Proceedings of the 1996 USENIX Technical Conference, San Diego, CA January 1996, 141-152.

[46] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, “BGP routing stability of popular destinations,” in Proc. ACM IMW, 2002.

[47] A.Detti. M. Pomposini, N. Blefari-Melazzi, S. Salsano, “Supporting the Web with an Information Centric Network that Routes by Name – Extended version”, available at http://netgroup.uniroma2.it/Andrea_Detti/Lookup-and-Cache/comnet-web-ICN-ext.pdf

[48] V. Jacobson, D. K. Smetters, N. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, R. Braynard, “VoCCN: voice over content-centric networks”, Proceedings of the 2009 Workshop on Re-architecting the Internet (ReArch 2009).

Appendix 1 – Computation of the the route inactivity timeout

The computation of the route inactivity timeout (TO) requires five variables: the last arrival time (LAT), the Smoothed Inter-arrival Time (SIT), the Inter-arrival Time VARiation (ITVAR), the Message Counter (MC), and the Maximum Inter-arrival Time (MIT). Using these variables, the value of the Timeout is computed with the usual TCP time out formula [25], with a multiplicative factor of 2, to avoid the competition of transport level congestion control and network level timeout, i.e. :

```
TO = 2 * (SIT + max (G, K*ITVAR))
```

where K=4 and G = 500 ms.

Let us now face the run-time computation of SIT and ITVAR. When the first Interest message of an inactive route or of a route not contained in the FIB is received, the variables are initialized as follow:

```
LAT = now

SIT = ISIT

ITVAR = 0

MC = 1

MIT = 0
```

where ISIT is a default initial value for SIT. We used ISIT = 1s in our test.

When the second Interest message of an active-route arrives, the variables are updated as follow:

```
MIT = now-LAT

SIT = MIT

ITVAR = SIT/2

LAT = now

MC = 2
```

When subsequent Interest messages of an active-route are received, the variables are updated as follow:

```
If PC < MCTh
```

```
        MIT = max(MIT, now-LAT)


        LAT = now
        MC = MC + 1
Else
        MIT = max(MIT, now-LAT)
        ITVAR = (1 - beta) * ITVAR + beta * |SIT - MIT|
        SIT = (1 - alpha) * SIT + alpha * MIT
        LAT = now
        MC = 0
```

where alpha=1/8, beta=1/4 and MCth = 16.

We are using the TCP algorithm to update SIT and ITVAR. However, in TCP the algorithm is fed with RTT samples, while our algorithm is fed with the maximum inter-arrival time (MIT) that we measure during a window of MCth packets. The rationale underlying the use of the window is that, when RTTs are long, the transport level congestion control generates bursty traffic and the throughput is RTT-limited. In these cases, the inactivity timeout has to be configured so as to not elapse during the empty time between consecutive bursts. Following this observation, we trust that, during a window of MCth, there are at least two bursts. In this case, the maximum measured value of the inter-arrival time is exactly the empty time between consecutive bursts, which is used to feed the computation of SIT and ITVAR.

Appendix 2 – Feasibility check taking into account web hosting

In the feasibility check carried out in Section 6, we assumed a *one-to-one* relationship between an IP address and a domain-name. As a consequence, we assumed that the number of ICN active-routes (N_{icn}) is equal to the number of current IP /32 active-routes (N_{ip}) toward Web servers, i.e. TCP port 80.

However, a more rigorous approach would have been to assume N_{icn} as equal to the number of current active-routes towards *domain-names*, rather than towards /32 IP addresses. Indeed, we envisage an ICN scenario where the routed name-prefixes are equal to the current domain-names (see Section 3).

We observe that, in case of Web Hosting services where the same IP address serves several domain-names, the assumed one-to-one relationship between an IP address and a domain-name *underestimates* the number of ICN active-routes. We used this assumption because we only have IP traces with anonymized IP addresses, which do not include domain-names of HTTP GET messages.

In this appendix we present a simulation analysis that takes into account Web Hosting services.

Our results show that, also in this more real scenario, the Lookup-and-Cache architecture is feasible by using current technology.

Lower and upper bounds - Since the same IP address may serve more than one domain-name, N_{ip} is a lower-bound of N_{icn}, which is reached when all flows that form an IP /32 active-route refer to the same domain-name. Furthermore, the average number of active-flows (N_f) is an upper-bound of N_{icn}, which is reached when all flows that form a IP /32 active-route refer to a different domain-name.

The inter-arrival time between IP /32 active-routes (I_{ip}) is an upper-bound of the inter-arrival time between ICN active-routes (I_{icn}). The inter-arrival time between active-flows (I_f) is a lower-bound of the inter-arrival time between ICN active-routes.

Tab. 2 reports the lower and upper bounds as measured in the traces. Results show that the difference between lower and upper bounds is much greater for tier-1 nodes (e.g., Equinix-sanjose-dirA) than for

tier-3 nodes (e.g. Rome-Tor-Vergata). Indeed, for tier-1 nodes, the probability of *concurrently* having many flows for the same IP destination is significant and so is the aggregation from active-flows (N_f) to IP /32 active-routes (N_{ip}). Conversely, this probability is quite negligible for tier-3 nodes where, often, an IP /32 active-route is formed by a single flow; hence, the average number of active-flows (N_f) is close to the average number of IP /32 active-routes (N_{ip}).

In what follows we carry out a simulation analysis aimed at estimating the values of N_{icn} and I_{icn}.

Trace id	Lower bound of average n. of active-routes (N_{ip}), values equal to ones of Tab. 1	Sim. average value of ICN active-routes (N_{icn})	Upper bound of average n. of active-routes (N_f)
Equinix-sanjose-dirA	2582	4680	28095
Equinix-sanjose-dirB	1293	1782	9766
Equinix-chicago-dirB	1197	1576	4265
Mawi-1	236	250	308
Mawi-2	246	267	303
Rome-Tor-Vergata	176	185	243

Trace id	Lower bound of average ICN active-routes inter-arrival (I_f)	Sim. average ICN active-routes inter-arrival (I_{icn})	Upper bound of average active-routes inter-arrival (I_{ip}). values equal to ones of Tab. 1
Equinix-sanjose-dirA	0.02 ms	0.5 ms	1.2 ms
Equinix-sanjose-dirB	0.09 ms	1.1 ms	2.8 ms
Equinix-chicago-dirB	0.17 ms	1.2 ms	1.7 ms
Mawi-1	1.7 ms	4.5 ms	5.9 ms
Mawi-2	1.8 ms	3.3 ms	4.1 ms
Rome-Tor-Vergata	2.3 ms	5.6 ms	7.3 ms

Tab. 2 – Bounds from Internet trace analysis and simulation results (domain-name Zipf alpha=1)

Simulation assessment – To measure N_{icn} from an IP packet trace, we need to know the domain-name addressed by each HTTP flow. Usually this information can not be extracted from publicly available traffic traces, since:

1) IP address are anonymized
2) The IP payload is limited to the transport header (UDP/IP), and does not contain HTTP information.

This is exactly what occurs for the traces [12][13][37], which we used for our feasibility check.

To derive N_{icn} in these conditions, we use a simulation model that associates to each HTTP flow a domain-name, singled out from the 1 million most used domain-names [43]. To do so, we *randomly* associate anonymous IP addresses to domain-names, statistically taking into account that an IP address may serve more than one domain-name.

The simulation model is depicted in Fig. 21. Briefly, for a given anonymous trace, we randomly associate the web servers' anonymous IP addresses of the trace to a set of public IP addresses, derived (as described below) from the 1 million most used domain-names. Then, we associate each anonymous flow of the trace to a domain-name, randomly extracted among those domain-names that have the public IP address associated with the web server's anonymous IP address of the flow.

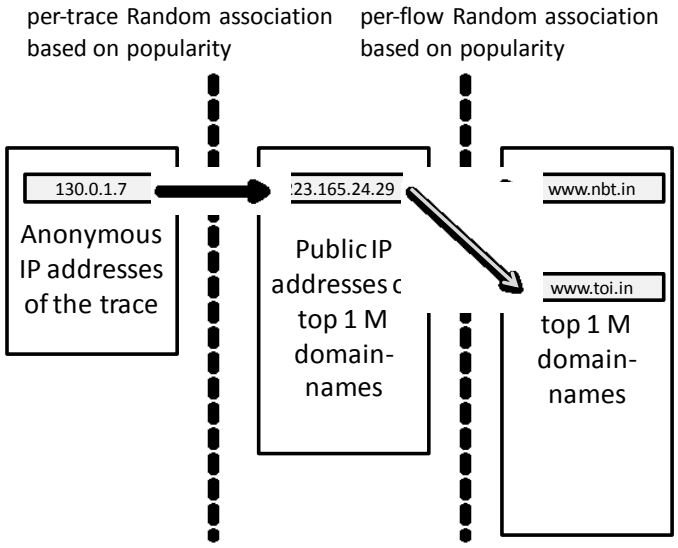

Fig. 21 – Simulation model to associate a anonymous IP address to a actual domain-name

More in details, the simulation model is formed by three phases, as follows:

Phase-1: data structures setup

1) We collect the top 1 million domain-names in a list named $\{DN\}$;
2) for each domain-name DN_i, we model its occurrence probability $opDN_i$ in an Internet trace as a function of its rank position, and according to a Zipf's law. Following the results of [44], we set the value of the Zipf alpha parameter to 1 [(14)];
3) we resolve the list $\{PubIP\}$ of the public IP addresses associated to each domain-name [15] (from a machine located in the campus of University of Rome Tor Vergata);
4) for each element $PubIP_i$, we compute its occurrence probability $opPubIP_i$ as the sum of the occurrence probability $opDN_j$ of the domain-names that use the IP address $PubIP_i$;
5) from the analyzed anonymized trace, we extract the list $\{AnIP\}$ of unique anonymized IP addresses of web servers;
6) for each element $AnIP_i$, we compute its occurrence probability $opAnIP_i$ as the ratio between the number of HTTP flows that have $AnIP_i$ as destination address and the total number of HTTP flows of the trace.

Phase-2: Random association of anonymous IP addresses to public IP addresses

7) since the number of public IP addresses $\{PubIP\}$ is in the order of 580k while the number of anonymous IP addresses of our trace is lower, we randomly extract a subset of public IP addresses, by using their occurrence probability $\{opPubIP\}$. We refer to this restricted set as $\{rPubIP\}$.
8) we map, one-to-one, elements of $\{AnIP\}$ to elements of $\{rPubIP\}$. We preventively sorted the elements of $\{AnIP\}$ and of $\{rPubIP\}$ on the base of the occurrence probabilities of their

[14] We remind that we are considering the occurrence distribution of domain-names rather than that of specific contents and that the parameter alpha of the domain-name Zipf [44] is greater than the one of the content Zipf [35] (e.g., 0.6, 0.8).

[15] Since the same IP address may serve several domain-names, the number of unique elements of $\{PubIP\}$ is lower than the length of $\{DN\}$. In our case the ratio between the length of $\{DN\}$ and the number of unique elements of $\{PubIP\}$ is equal to about 1.7.

elements. Consequently, the element of $AnIP_k$ with rank k in terms of occurrence probability is mapped to the element $rPubIP_k$ that has the same rank.

Phase-3: Random association of anonymous flows to domain-names

9) for each flow of the trace, we map its destination anonymous IP address $AnIP_i$, to the public address $rPubIP_i$ and we randomly associate to it a domain-name randomly extracted among the ones that use $rPubIP_i$. The extraction is properly weighted by the occurrence probability $opDN_i$.

Since each flow has now an associated domain-name, we can evaluate the number of ICN active-routes and average active-route inter-time. Results are reported in Tab. 2. We observe that the expected value of N_{icn} and I_{icn} are rather close to the values of N_{ip} and I_{ip}. As discussed in section 6, these values are well supported by current technology; hence the Lookup-and-Cache architecture is feasible, even considering that an IP address may serve several domain-names.