Route discovery and caching: a way to improve the scalability of Information-Centric Networking

N. Blefari Melazzi, A. Detti, M. Pomposini, S. Salsano
Department of Electronic Engineering
University of Rome, Tor Vergata, Roma, Italy
{blefari, andrea.detti, matteo.pomposini, 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. In this paper, we first describe what, we believe, are the main advantages and components of an ICN infrastructure; then we present an overall architecture for ICN, and then we focus on the main contribution of the paper, which is a route caching technique, designed to improve the scalability of the routing by name functionality.**

Keywords: Internet Architecture, Future Internet, Information-Centric Networking, Routing, Caching, Sacalability

I. 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] 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; this copy could be stored in the original server, in a cache contained in a network node or even in another user's device; iii) deliver the content back to the requesting host.

ICN can be seen as an evolution of network switching modes, from circuit switching through packet switching to content switching. In circuit switching, a PCM slot contains only user data; in packet switching an IP datagram contains (among other things) destination addresses and pieces of user data; in "content switching" data units should contain (almost) everything: a package of user data, signalling information, meta-data describing the content and how to handle it, and security information. As a result, in our view, an ICN would offer the following advantages:

i) efficient content-routing. Even though today's 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;

ii) in-network caching. Caching enabled today by off-the-shelf HTTP transparent proxies requires performing stateful operations. 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, especially for video;

iii) simplified support for peer-to-peer like communications, without the need of overlay dedicated systems. Users could obtain desired contents from other users (or from caching nodes) thanks to content-routing and forward-by-name 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;

iv) simplified handling of mobile and multicast communications. 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, without the need of overlay mechanisms;

v) 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.

vi) content-oriented quality of service differentiation (and possibly pricing), providing different performance in terms of both transmission and caching. Network operators (especially mobile ones) are already trying to differentiate quality and

This work is partially founded by the EU (FP7 CONVERGENCE project)

priority of content, but they are forced to resort to complex deep packet inspection technologies. ICN would let operators differentiate the quality perceived by different services without complex, high-layer procedures [12], and off-load their networks via caching, a very handy functionality, particularly for mobile operators who can differentiate quality and priority of content transferred over the precious radio real estate;

vii) content-oriented access control, providing access to specific information items as a function of time, place (e.g. country), or profile of user requesting the item. This functionality also allows implementing digital forgetting, to ensure that content generated at one period in a user's life does not come back to haunt the user later on, and garbage collection, deleting from the network expired information;

viii) possibility to create, deliver and consume contents in a modular and personalized way;

ix) network awareness of transferred content, allowing network operators to better control information and related revenues flows, favoring competition between operators in the inter-domain market and better balancing the equilibrium of power towards over the top players;

x) 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, social gatherings, mobile networks on board vehicles, trains, planes). This last point is maybe the most important one, especially to stimulate early take up of ICN in selected (and possibly isolated) environments.

On the cons side, ICN has some drawbacks and challenges. A first, obvious, con is that it requires changes in the basic network operation. 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 rest of this paper, we first recall an architecture that we proposed in the CONVERGENCE project [11] and in [13] and then we propose our main contribution, which is a route caching technique that improves scalability.

II. Overall Architecture

We propose an architecture called CONET (COntent NETwork) [13], which is defined as an inter-network that connects CONET Sub Systems (CSSs) (see Figure 1). A CSS contains CONET nodes and exploits an under-CONET technology to transfer data among CONET nodes. The devices within a CSS can use an autonomous and homogeneous under-CONET addressing space and an interior under-CONET routing protocol. A CSS could be: 1) a couple of nodes connected by a point-to-point or an overlay link, like the CSS n.1 of Figure 1; 2) a layer 2 network like Ethernet, like the CSS n.3 of Figure 1; 3) a layer 3 network, e.g. a private IPv4/IPv6 network or a IPv4/IPv6 subnet or a whole Autonomous System or even the whole current Internet, like the CSS n.2 of Figure 1.

CONET nodes exchange CONET Information Units (CIUs): *interest* CIUs convey requests of named-data; *named-data* CIUs transport chunks of named-data, e.g., parts of a file. To best fit the transfer units of an under-CONET technology, all CIUs are carried in smaller CONET data units named carrier-packets.

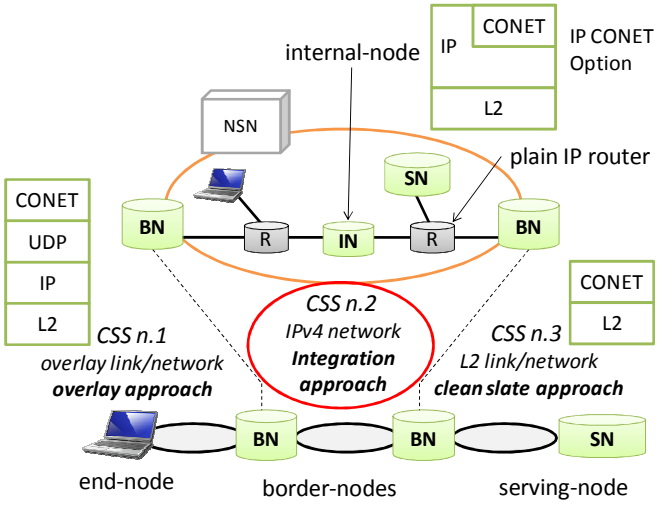

Figure 1. CONET Architecture

CONET nodes are classified as end-nodes (ENs), serving-nodes (SNs), border-nodes (BNs), internal-nodes (INs) and name-routing-system nodes (NRSs). End-nodes are user devices that request named-data by issuing interest CIUs. Serving-nodes store, advertise and provide named-data by splitting the related sequence of bytes in one or more named-data CIUs, which are transferred by means of carrier-packets. Border-nodes, located at the border between CSSs, forward carrier-packets by using CONET routing mechanisms (i.e. taking into account the requested content-name: routing-by-name) and cache named-data CIUs. Optional Internal-Nodes could be deployed inside a CSS to provide in-network caches; differently from border-nodes, internal-nodes forward carrier-packets by using only under-CONET routing mechanisms. Name-Routing-System nodes are used in a CSS to assist the CONET routing-by-name process. As shown in Figure 1, Border Nodes interconnect different CSSs, therefore the end-to-end forward-by-name process can be seen as the process of finding a sequence of Border Nodes from an End-Node up to a Serving Node. In short, CONET is an interworking protocol, just like IP.

The operation in a CONET internetwork can be described as follows. A Border Node checks if the content requested is available in its cache; if not it performs forward-by-name. If the CSS is an IP network, the result of the forward-by-name operation is the IP address of the upstream Border Node, therefore the content request can be sent using this destination IP address. An Internal Node in the path between the two border nodes "intercepts" the content request, it checks if the requested content is available in its cache, if not it forwards the packet using the IP destination address. A plain IP Router in

the path between the two Border Nodes will simply forward the packet looking at the IP destination address. When data packets providing the requested content are generated by the Serving Node towards the End-node (or by any Border or Intermediate Node that had cached the content), the crossed downstream Border Nodes and Internal Node can in turn cache the content while forwarding it. In this way, further content requests for the same content will not need to travel up to the Server Node.

The three typologies of CSSs depicted in Figure 1 correspond to different deployment scenarios: i) overlay: CONET on top of the IP layer, as it occurs in the CSS n.1 of Figure 1; ii) clean slate: CONET on top of layer-2, as it occurs in the CSS n.3 of Figure 1; iii) *integration*: CONET integrated in the IP layer, as it occurs in the CSS n.2 of Figure 1.

The first two approaches are known in the literature. The integration approach supports CONET in a CSS that is an IP network (IP-CSS). Depending on where CONET routing protocols are deployed (i.e. where we deploy Border Nodes) we have different scenarios: if CONET protocols are implemented only in user equipments, interconnected by the current Internet, then we have only one CSS: the current Internet. If they are implemented in current border gateways (i.e. where BGP runs), then CSSs coincide with current Autonomous Systems. If they are implemented in all current routers, then CSSs coincide with current IP subnets.

Additionally (but optionally), we propose to make IP itself content-aware, by transporting the identifier (name) of a CONET carrier-packet in a novel IPv4 or IPv6 option, which we name CONET option [15]. The advantages of this approach with respect to the overlay one is that it allows nodes to quickly forward carrier-packets, without the need of terminating upper layer protocols or performing a "deep packet inspection". This is a major requirement to deploy information-centric features in nodes where a high packet rate demands a fast forwarding operation. In addition, this approach allows deploying CONET routing-by-name functions only in a subset of nodes (i.e. Border-nodes and End-nodes) while allowing caching in all nodes running the new IP option (i.e. Internal nodes). On the contrary, in the overlay approach, caching in all nodes would require to deploy routing-by-name functionality in all nodes. The disadvantage is that we require a new IP option, but this is much less disruptive than the clean-slate approach.

To transform this conceptual architecture into a fully fledged system we need to define (at least) the following fundamental components:

i) primitives & interfaces, which define the relationship of the ICN protocols with the overall architecture

ii) the naming scheme, which specifies the identifiers for the contents addressed by ICN.

iii) the forward-by-name (or route-by-name) mechanism, used by ICN nodes to relay an incoming content request to an output interface. The output interface is chosen by looking up a "name-based" forwarding table

iv) the routing protocols used to disseminate information about location of contents, so as to properly setup the name-based forwarding tables

v) the data forwarding mechanism that allows the content to be sent back to the device that issued a content request. Data forwarding cannot use the forward-by-name mechanisms, because, typically, devices/interfaces are not addressed by the content routing plane of an ICN

vi) in-network caching, which concerns the ability of ICN nodes to cache data and to reply to incoming content requests

vii) segmentation & transport mechanisms (see e.g. [14]) needed to: 1) split a content in different chunks (each chunk is an autonomous data unit with embedded security and addressable by the routing plane); ii) ensure a reliable transfer of chunks from the origin node (or from a cache node) towards the requesting node; iii) counteract congestion

viii) security & privacy issues tackling (at least) three specific aspects: 1) how to guarantee content authenticity and 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.

In this paper we focus on the forwarding and routing components.

III. ROUTE DISCOVERY AND CACHING AND RELATED WORK

In our scenario, we may need to handle tens of billions of name-based routes, due to the high numbers of possible 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 possible ICN routes; second, realizing a so large Routing Information Base (RIB) requires a costly hardware. To overcome these problems, we propose a routing-by-name functionality, named Lookup-and-Cache, where the FIB of a node is used as *cache of routes*, while 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 [16], the Lookup-and-Cache architecture seems feasible since the number of routes *concurrently* needed by a node to forward traffic is rather small and lower than the capacity provided by current FIB technology (see below).

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 a proposal of an architecture that puts together these two techniques. In 1988, Feldmeier [17] 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 decoupling between the FIB and the RIB imposes a careful rethinking of route replacement algorithms. In [18] the Authors revised route caching as an instrument to face the growing of IP routing tables with limited FIB memories. As in [17], they supposed to have a local RIB. In [19], the authors proposed to separate IP routing from routers. The routers would simply forward packets while a centralized Routing Engine would select routes on behalf of the IP routers in each AS and exchange reachability information with other Routing Engine

of other autonomous systems (AS). 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 architecture, the FIBs *pulls* the routes from the Routing Engine. Separating packet forwarding from control decisions is also at the basis of the SDN (Software Defined Networking) paradigm [20], which has been proposed quite recently. Our Lookup-and-Cache does not require SDN to be implemented, even if the node-to-NRS interface could be remapped into the OpenFlow [20] switch-to-controller API.

IV. WORKING ASSUMTPIONS

Before presenting the details of our solution we introduce some working assumptions. Our solution can be of course applied to our own architecture but also in other ICN systems that route-by-name content requests such as CCN [3] [4].

A. 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 discover and deliver named chunks. In order to fetch a chunk, a user issues a data unit, named *interest CIU* or simply *Interest*, that contains the name of the chunk. ICN nodes *route-by-name* the Interest, 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 in a named-data CIU, or simply *Data*. Network nodes forward Data towards the client, through the same sequence of ICN nodes previously traversed by the Interest message. The Data forwarding process exploits reverse-path information either temporarily leaved in the traversed nodes during the Interest forwarding process (see Pending Interest Table of [3]), or contained in the header of Data message, as previously collected in the Interest message during its forwarding process (see reverse-path source-routing in [13]). 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 messages is regulated by a receiver-centric congestion control mechanism [17] [21], which could be based on the same logic used by TCP.

B. Naming scheme

As regards the naming scheme, several proposals (e.g. [2][3][4][22]) agree in adopting a hierarchical naming. Here we assume a rather general hierarchical naming scheme where a name is formed by a sequence of *Components*: a name has the form "Component_1/Component_2/../Component_n". This scheme supports current Web URL, where 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. A hierarchical naming is also able to support human readable names [3][4] and self-certifying names [2][22] (where Component_1 is the *Principal* and Component_2 is the *Label*). The full sequence of Components is referred to as the *chunk-name*.

C. Number of ICN routes

We assess the number of possible ICN routes by assuming that: i) the ICN network will serve current 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, a node is within the "default-free" zone of the network, i.e. it does not use a default route. With these assumptions, in [23] we conclude that the expected number of ICN routes a node should handle is close to the number of name-prefixes advertised by ICN servers, equal to about 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>.

If we change the assumptions stated above, these numbers would obviously change. 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. In case of a node that has a default route, e.g. corresponding to a tier-2 or a tier-3 node in current Internet, the number of ICN route might be radically lower than 10^9, and so forth.

V. THE LOOKUP-AND-CACHE ROUTING ARCHITECTURE

In [23] we show that if we used the current IP router architecture to support ICN, under the above assumptions, the size of current FIBs should be increased by a factor of 10^3, while RIBs should be larger by a factor of 10^2. To avoid this (significant) increase of capacity and cost, we propose our *Lookup-and-Cache* architecture, which uses the FIB of a Forwarding Engine as a *route cache* and exploits a *centralized routing engine* that serves all the nodes of a sub-system.

Figure 2 reports an example of the 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, looks up the route in a remote RIB, inserts the replied information in the FIB, de-queues and forwards the Interest message. In what follows, we discuss the rationale underlying the Lookup-and-Cache architecture and its main components.

A. FIB as a route cache

It is argued that the relative frequency with which Web contents are requested follows the Zipf's law [16]. Therefore, a many *flows* of Interest messages that an ICN node has to concurrently route-by-name refer to a small set of contents. More important, these flows use an even smaller set of ICN

routes, since ICN routes address servers rather than single contents (see below). In [23] we show that this set of, so called, *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. When the FIB lacks a route, the node lookups the route in a "remote" RIB and then caches the route in the FIB. When all FIB rows are filled in, new routing entries may substitute old ones, according to a route replacement algorithm.

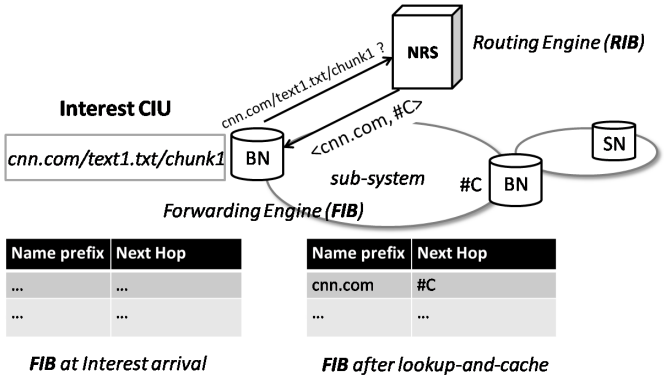

Figure 2. Lookup-and-Cache concept

B. Centralized Routing Engine

All the ICN routes are contained in the RIB of a Routing Engine, which serves all nodes of a CONET Sub-System (CSS) and runs on a centralized server, named *Name Routing System* (NRS) node. Thus, the cost of an expensive Routing Engine able to handle 10^{10} of routes is taken only for a single network device, rather than for all network nodes. Furthermore, since many Interest flows use a small set of active-routes, the temporal dynamics of active-routes is *slower* than the flow ones. Indeed, a route is used for a period of time that is greater than or, at least equal to, the duration of a single flow. This limits the lookup rate that a centralized Routing Engine should deal with; below we show that this rate is easily supported by current database technologies.

So far we have described the "forwarding-plane" of the Lookup-and-Cache architecture. The routing-plane, i.e. how NRS nodes disseminate name-prefixes in order to properly setup their RIB, is described in [23]. However, our proposal is simply to adapt and reuse the REGISTER and UNREGISTER functions of the DONA architecture, whose feasibility and performances have been already fully discussed in [2].

C. 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: i) if the FIB is not full, the node performs a lookup in the remote RIB and store the new route in the FIB; ii) the forwarding of the Interest messages is subject to a route-lookup delay. When the FIB is full, the insertion of a new route implies the replacement of an old route. In this case, a *route replacement algorithm* decides whether to lookup the new route or not. In the first case it also decides which old route has to be replaced. In the second case, the Interest message is dropped and subsequently retransmitted by transport level mechanisms.

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 inconveniences, it would be desirable to replace *inactive* routes. Consequently, the design of the route replacement algorithm aims at solving two problems: first, how to identify *inactive* routes 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. We propose a route replacement algorithm, based on the estimation of an Inactivity Time Out (ITO); its performance are compared with the Least Recently Used (LRU) policy. For lack of space, further, important details on our route replacement algorithm are reported in [23] and include: consistency issue of cached routes in FIB, invalidation mechanisms, and a more complete performance evaluation.

How to identify inactive routers - The ITO algorithm assumes that each route contained in the FIB has an inactivity time out (ITO), after which the route is considered inactive. The timeout value is calculated by the same algorithm used for the TCP retransmission time out, where, instead of using round-trip-time measurements as inputs, we use measurements of inter-arrival times between two consecutive Interest messages of the same route.

How to behave in case of FIB overload – The ITO algorithm is *non-preemptive*, i.e. an active-route cannot be removed from the FIB. When a new route needs to be inserted there are two options: i) in presence of inactive-routes (*FIB underload*), the least recently used inactive route is replaced by the new route; ii) in absence of inactive-routes (*FIB overload*), the new route is not inserted, and the incoming Interest message is discarded. The *non-preemptive* approach needs to be carefully evaluated, because it may prevent the forwarding of traffic. Nevertheless, in case of route overload the *non-preemptive* approach avoids in/out flapping of routes in the FIB. In/out flapping is harmful, since it overwhelms the NRS node, increases the time required by users to download contents, and disfavours long downloads.

VI. NUMBER OF ACTIVE ROUTES AND LOOKUP RATE

In this section we report our findings about expected number of active-routes and lookup rate, to check if these figures are compatible with current technology.

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. 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-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, the couple < IP destination address, destination Port> contained in TCP ACK messages would be replaced by a chunk-name that contains the name-prefix advertised by the Web server. As a consequence, we can

"remap" a trace of TCP ACKs, captured from the current Internet, in an "equivalent" trace of ICN Interest messages, as it would be generated in a "real" ICN. This we did in [23] for several real traces. In the worst case (Equinix-sanjose-dirA trace, of a tier-1 node [23]), the maximum number of active-routes is in the order of 10^3; such a value is much lower (by a factor of 10^3) than the capacity provided by of an off-the-shelf SRAM-based FIB. This small value is the result of the Zipf nature of the Web [16], for which a wide set of flows refer to a limited set of "popular" contents and use an even more limited set of ICN routes. In facts, a flow-level analysis revealed that the whole trace contains about 2.6 millions of Interest flows, i.e. of content downloads. Nevertheless, these 2.6 millions of Interest flows use only 11000 routes.

As regards the lookup rate, an analysis of the same traces shows that the average inter-arrival time between the start of two consecutive active-routes is in the order of milliseconds. When the FIB memory is properly dimensioned for containing all active-routes, the inverse of the active-routes inter-arrival time represents an upper bound of the lookup rate. Indeed, we have a lookup at the start of the route activity only if the route is not already cached in the FIB. Therefore, an average active-route inter-arrival time in the order of few ms implies, in the worst case, a lookup rate in the order of 1000 lookups per second, which is by far supported by current database technology. For instance, we implemented the functionality of the NRS node with a Bind9 server, running on a Linux laptop with an Intel Pentium Processor M at 1.4 Ghz and, even using this dated hardware, we measured a sustainable rate of about 15000 lookups per second.

We investigated also the effectiveness of FIB *over-provisioning*, to reduce the lookup rate. We say that a FIB is over-provisioned, when it has a capacity greater than the maximum number of expected active-routes. We observed a significant decrease of the lookup rate as the FIB size increases. This occurs because each route frequently switches on and off, during the whole trace. For instance, the Equinix-sanjose-dirA trace contains a number of route activations/deactivations equal about to 54000, whereas the number of unique routes is about 11000. Due to this strong temporal correlation among route activities, the increase of the FIB size has a relevant impact on the cache hit-ratio and, hence, on the lookup rate. When the FIB size increases from the minimum value of 2700 (required to operate in *over-provisioning* conditions*)* to about 9000 entries, the lookup rate decreases from 1000 to 200 lookups per second.

It is important to note that we assumed a one-to-one relationship between an IP address and a domain-name. In Appendix II of [23], we present a simulation model that takes into account Web Hosting services. 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, and our conclusions remain valid.

VII. TEST-BED ANALYSIS

In Figure 3 we show a test-bed that we used to demonstrate our solution. We have two sub-systems; each sub-system is an IP network connected by a 100 Mbit/s Ethernet switch. Sub-system A contains three ICN clients. Sub-system B contains an ICN server and an NRS node. The sub-systems are interconnected by a node *BN*, equipped with two network interfaces. All devices run the Linux OS.

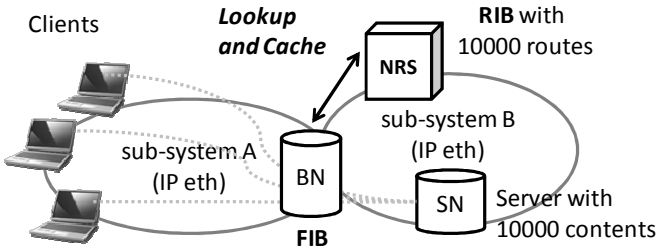

Figure 3. Testbed setup

The FIBs of the clients have a single default route toward node *BN*; hence, a client does not perform Lookup-and-Cache procedures. Node *BN* does not have a default route, but uses the Lookup-and-Cache mechanism to feed its FIB, up to a fixed size of 100 routing entries. The server has a repository containing 10000 contents, the content size follows a Pareto distribution ($k = 133k$, $\alpha = 1.1$) and each content is divided in chunks of 4 kBytes. The NRS node has a RIB indexing all the 10000 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 a Bind-9 DNS server. To emulate the delay of a wide area network, we artificially set up a two-ways delay of 100ms 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 *local* lookup-and-cache procedures, such as inserting the route in the FIB, de-queuing the waiting Interest message, 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.

We analyze the performance of Lookup-and-Cache as a function of the number of active-routes. The workload generates a constant number of active-routes, by keeping fixed the number of concurrent downloads performed by clients. The reason for this choice is that the number of active-routes is a crucial variable to highlight performance limits of the Lookup-and-Cache routing. 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 is started. In case the first Interest message is dropped, the message is periodically resent each 2 seconds. This delay is duly taken into account in the performance evaluation and in the comparison with LRU.

We measured the performance with a FIB of 100 entries, by using both ITO and LRU caching replacement policies. As a benchmark, we compare these performances with the case of an *unlimited* FIB, where all routes are properly preloaded. Each test is repeated 5 times and we measured both average performances and 95% confidence intervals. Figure 4 reports the average download time. Figure 5 reports the average number of lookups per download, that is the ratio between the number of lookups performed by node *N* and the overall

number of downloads. We first analyze the cases where the FIB is under-loaded, i.e. the number of active-routes is lower than 100, and then the cases where the FIB is overloaded.

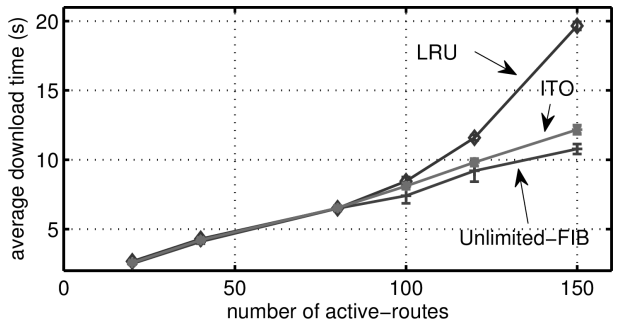

Figure 4. Average download time versus number of active-routes

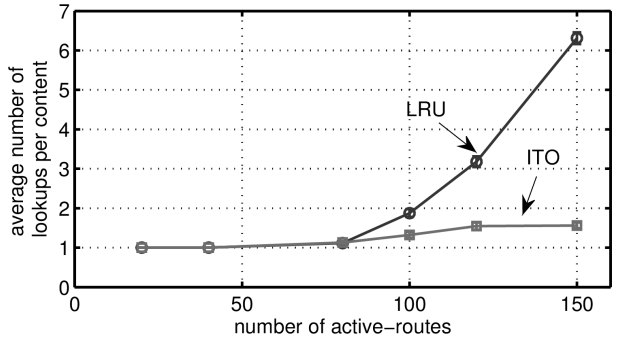

Figure 5. Lookup rate versus number of active-routes

Underloaded FIB. Lookup-and-Cache works well when the number of active-routes is lower than the FIB size (100 routes). In these conditions, 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 correctly held in the FIB for the download time. The download time is composed of the initial lookup delay (160 ms, if the route is not already in the cache) plus the content download time. The latter depends on the sharing of the available link capacity among the downloads, therefore it linearly increases with the number of downloads (active routes). In these underloaded conditions, ITO and LRU do not show noticeable differences: both algorithms succeed in replacing inactive routes.

Overloaded FIB. Performances start to decrease when the number of active-routes gets close to, or overcomes, the FIB size. In these conditions, the greater the overloading, the greater the performance degradation with respect to the unlimited-FIB case. In addition, the differences between the route replacement algorithms shows up. With the ITO algorithm, the download delay is a bit greater than the unlimited-FIB case and the number of lookups per download remains quite limited. We argue that these promising results are the consequence of the *non-preemptive* policy adopted by ITO algorithm. Figure 5 shows that the ITO algorithm limits the number of lookups per download to 1.5, also in overloaded conditions. The rise of the lookup rate is due to the presence of 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 briefly reinserted; this increases the number of lookup per content. With the LRU algorithm, we observe a significant increase of the number of lookups per download and longer download times. When the number of active routes gets close to and overcomes the FIB size, the LRU algorithm is *pre-emptive*, as it replaces FIB entries associated to active routes. This yields a dramatic in/out flapping of the routes in the FIB, which increases download delay and number of lookups per content.

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, A. Ermolinskiy, Kye Hyun Kim, S. Shenker, I. Stoica: “A data-oriented (and beyond) network architecture”, Proc. of ACM SIGCOMM 2007

[3] V. Jacobson, et al., ”Networking named content”, in Proc. of 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] SAIL project website, http://www.sail-project.eu/

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

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

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

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

[11] CONVERGENCE project website: www.ict-convergence.eu

[12] S. Oueslati, J. Roberts, N. Sbihi: “Ideas on Traffic Management in CCN”, Information-Centric Networking, Dagstuhl Seminar

[13] A. Detti, N. Blefari-Melazzi, S. Salsano, M. Pomposini, “CONET: A Content Centric Inter-Networking Architecture”, ACM SIGCOMM Workshop on Information-Centric Networking, Toronto, August 2011

[14] 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

[15] A. Detti, S. Salsano, N. Blefari-Melazzi, “IPv4 and IPv6 Options to support Information Centric Networking”, Internet Draft, draft-detti-conet-ip-option-02, Work in progress, October 2011

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

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

[18] 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

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

[20] N. McKeown et al. “OpenFlow: Enabling Innovation in Campus Networks”, ACM SIGCOMM Computer Communication Review, Volume 38, Number 2, April 2008

[21] 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

[22] 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

[23] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, “Supporting the Web with an Information Centric Network that Routes by Name”, Tech. report, available at http://netgroup.uniroma2.it/Andrea_Detti/Lookup-and-Cache/tech-rep-ICN.pdf