CONET: A Content Centric Inter-Networking Architecture

A. Detti, N. Blefari-Melazzi, S. Salsano, M. Pomposini
Department of Electronic Engineering, University of Rome "Tor Vergata"
Via del Politecnico 1, Rome (Italy)

{andrea.detti, blefari, stefano.salsano, matteo.pomposini}@uniroma2.it

ABSTRACT
CONET is a content-centric inter-network that provides users with a network access to remote named-resources, rather than to remote hosts. Named-resources can be either data (named-data) or service-access-points (named-sap), identified by a network-identifier (a name). CONET interconnects CONET Sub Systems, which can be layer-2 networks, layer-3 networks or couples of nodes connected by a point-to-point link. CONET supports the already proposed "clean-slate" and "overlay" deployment approaches. In addition, CONET supports a novel "integration" approach, which extends the IP layer with a new header option that makes IP itself content-aware. CONET limits the size of name-based routing tables by including only a subset of all named-resources; missing entries are looked up in a name-system and then cached. CONET does not maintain states in network nodes, to deliver contents.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms
Algorithms, Design, Experimentation

Keywords
Internet architecture, content-centric networking, route-by-name, in-network caching, route caching, IP option.

1. INTRODUCTION
Several papers (e.g. [1][2][3][4]) and research projects ([5][6][7]) propose a shift from "host-centric networking" to "information centric" or "content-centric" networking. The essence of Content-Centric Networking (CCN; we will speak of CCN to denote the general trend) is that the network layer provides users with contents, instead of providing communication channels between hosts, and is aware of such contents, at least in the sense of knowing the "name" of the contents. A CCN architecture should:

- address contents, using an addressing scheme based on names, which do not include references to their location;
- route a user request, which includes a content-name, toward the closest copy of the content with such a name (name-based, anycast routing) and deliver the content to the requesting host;
- provide a native, in-network caching functionality to achieve efficient content delivery both in fixed and mobile environments [8];
- exploit security information embedded in the content to avoid the diffusion of fake versions of contents and to protect the content, as opposed to exploit connection-based or application-based security; protecting information at the source is more flexible and robust than delegating this function to applications, or securing only the communications channels [4];
- provide a way to differentiate the quality perceived by different services [9], and provide a per-content quality of service differentiation, including cache hits performance.

Among the advantages of CCN, discussed e.g. in [3][4], in this paper we focus on the improved and built-in support of a replication/caching functionality. Users should "retrieve desired content regardless of where it comes from – the original source, a copy on their local disk, or the user next to them in Starbucks^{TM}," [4].
It is true that content replication is already supported by CDNs [10], but CDNs are proprietary and closed facilities. It is also true that in-network caching is already supported by so called-transparent proxy technologies, but this is done at application level and requires a stateful tracking of user connections. Stateful procedures limit the application of caching in high-speed nodes, where a stateless CCN could instead recognize and cache contents on the fly.
On the cons side, CCN 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 obvious implications on the size of routing tables and on the complexity of lookup functions; ii) in some proposed CCN architectures [3], guaranteeing bidirectional communication (reverse paths) requires maintaining states in network nodes. This very argument was, maybe, too heavily used against the Integrated Services architecture (and the RSVP protocol) but it is surely an issue that deserves careful investigations.
When CCN is meant as a replacement of the current network layer, it poses the challenge of how to efficiently support communication sessions based on models different from content retrieval (e.g. http connections for e-commerce applications, instant messaging, social networking; rtp connections for real-time communications). These communication sessions rely on host addresses, and need suitable solutions to work in a CCN environment, as shown in [11] for SIP based VoIP applications.
The goal of this paper is to introduce a CCN architecture that tries to achieve the pros of CCN, and specifically a built-in caching functionality, while mitigating the cons.
Our CONET is an (inter-)network layer that provides users with a network access to remote *named-resources*, rather than to remote hosts.

ICN'11, August 19, 2011, Toronto, Ontario, Canada.

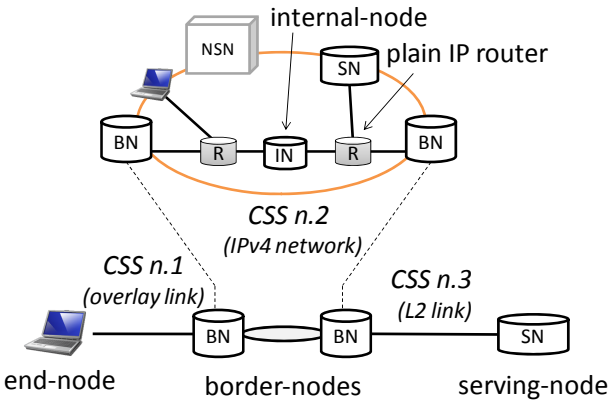

Fig. 1 - CONET Architecture

Named-resources can be data (*named-data*) or service-access-points (*named-sap*), identified by a network-identifier (NID). By default, the NID is an anycast address and CONET may contain multiple replicas of the same named-resource.
In the case of named-data (e.g. a file), CONET enables users to retrieve it by using the best set of networked devices (caches, mirror servers, etc.) that can provide that named-data. In the case of a named-sap (e.g. the logical port of a server), CONET provides the means to exchange point-to-point data between a requesting entity and an entity addressed by such named-sap. The named-sap case can be extended to multicast; in this case the NID of a named-sap has a multicast meaning, rather than an anycast one. The main features of CONET are:

- it is stateless: network nodes do not maintain information on the ongoing communications;
- it limits the size of name-based routing tables by caching only a subset of all possible routes; missing routing entries are looked up in a name-system and then cached;
- it can be integrated in the actual IP networks by using a new header option, which makes IP itself content-aware [12]. In this case, the nodes could use hybrid routing tables containing both IP network addresses and names. However, CONET also supports the traditional clean-slate or overlay deployment approaches.

In the following we present the details of our solution and substantiate these statements. For lack of space, we do not deal with security issues and we only consider the transfer of named-data, neglecting the use of CONET for named-sap (D3.1 in [7] provides further details on named-sap).

2. THE CONTENT INTER-NETWORK (CONET)

2.1 Network Architecture

CONET is a system that interconnects CONET Sub Systems (CSSs) (see Fig. 1). A CSS contains *CONET nodes* and exploits an *under-CONET* technology to transfer data among *CONET nodes*. A CSS could be:

- a couple of nodes interconnected by a point-to-point link, e.g. a PPP link or a UDP/IP overlay link;
- a layer-2 network, e.g. Ethernet, or a layer-3 network, e.g. a private/public IPv4 or IPv6 network, or a whole IP Autonomous System, or even the whole current Internet.

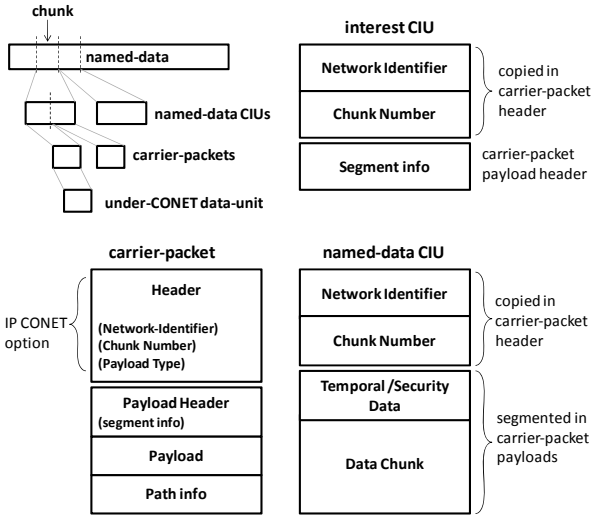

Fig. 2 –CONET Information Units (CIUs) and carrier-packets

The devices within a CSS use an autonomous and homogeneous under-CONET addressing space and, if necessary, an interior under-CONET routing protocol (e.g. [13]).
CSSs can be defined rather freely. For instance, if CONET protocols are implemented only in user equipments, interconnected by the current Internet, then we have only one CSS: the current Internet. If CONET protocols are implemented in current border gateways (i.e. where BGP runs), then CSSs coincide with current Autonomous Systems. If CONET protocols are implemented in all current routers, then CSSs coincide with current IP subnets. If CONET protocols are implemented in nodes that interconnect different layer 2 networks, removing IP, then CSSs coincide with such layer 2 networks.
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 (see Fig. 2). To best fit the transfer units of an under-CONET technology, all CIUs are carried in smaller CONET data units named *carrier-packets*.
CONET nodes are logically classified as end-nodes (ENs), serving-nodes (SNs), border-nodes (BNs), internal-nodes (INs) and name-system-nodes (NSNs). *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 (see again Fig. 2). *Border-nodes*, located at the border between CSSs, forward carrier-packets by using CONET routing mechanisms (i.e. routing-by-name and inter-CSS source-routing, described below) 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. Optional *Name-System-Nodes* are used in a CSS to assist the CONET routing-by-name process (see Sec. 3). CONET may be deployed following three approaches:

- *overlay approach*: CONET on top of the IP layer; CSSs are couples of nodes connected by overlay links, e.g. UDP/IP tunnels, as it occurs in the CSS n.1 of Fig. 1;
- *clean slate approach*: CONET on top of layer-2 technologies (e.g. Ethernet, PPP, MPLS LSP); CSSs are nodes connected by layer-2 links/networks, and CONET replaces the IP layer, as it occurs in the CSS n.3 of Fig. 1;

- *integration approach*: CONET functionality integrated in the IP layer by means of a novel IPv4 option [12] or by means of an IPv6 extension header, as it occurs in the CSS n.2 of Fig. 1.

While different variants of the clean-slate and overlay approaches have been already discussed in the literature [3], [5], [6], the proposed integration approach is novel, to the best of our knowledge; therefore in this paper we focus on this approach, describing it in Sec. 4. We also note that within our proposed architecture, the three approaches are not mutually exclusive, but they can be combined.

2.2 Model of operations

This section provides an example of CONET operation in the scenario depicted in Fig. 1, considering an end-node that retrieves a named-data from a serving-node. We assume that the routing information that enables to *reach-by-name* the named-data has been already distributed in the CONET. This process is initiated by the serving-node that advertises the related network-identifier by using a name-based routing protocol, as described in Sec. 3.
The retrieval of a named-data involves a sequence of a *request - delivery* phases in which the end-node requests and obtains named-data CIUs and then reassembles the whole named-data (Fig. 2). For simplicity, in the following we consider a case in which the named-data is fully contained in a single named-data CIU that, in turn, is fully contained in a single carrier-packet. Therefore, only one request-delivery phase is needed.

Request

- an end-node requests the named-data CIU by issuing an interest CIU, which includes the network-identifier of the named-data; the interest CIU is encapsulated in a carrier-packet, named *I*;
- the end-node and intermediate border-nodes *route-by-name* the packet *I*. The route-by-name process singles out the *CSS address* of the next border-node toward the serving-node, on the basis of the network-identifier contained in *I*. A CSS address is an address consistent with the traversed under-CONET technology (e.g., an IPv4 address). Then, the routing engine encapsulates the carrier-packet *I* in the under-CONET data-unit and uses the CSS address as the destination address;
- the CSS address of the end-node and the set of CSS addresses of the traversed interfaces of border-nodes in the "upward" path are appended, by these nodes, to the carrier-packet *I*, within a control field named *path-info*[1];
- the internal-nodes parse carrier-packet *I* and then forward it by using the under-CONET routing engine.

Delivery

- the first in-path CONET node (BN, IN or SN), which is able to provide the named-data CIU requested by *I*, will send back the CIU, without further propagating *I*;
- this named-data CIU is encapsulated in a carrier-packet, named *C*. The carrier-packet *C* traverses the same CSSs of the carrier-packet *I*, but in the downward direction and will reach the requesting end-node;
- the serving and the border nodes perform the *inter-CSS* reverse-path routing in a *source-routing* fashion, by using the path-info control field. This path-info is the copy of the one set up in *I* during the upward routing;
- within a CSS, the under-CONET technology (e.g. IP) performs the routing of carrier-packet *C*; therefore traditional traffic engineering mechanisms could be used;
- all border-nodes and internal-nodes in the downward path may cache the named-data CIU contained in *C*.

We observe that the use of inter-CSS source-routing on the reverse-path does not require to have "pending" states in the traversed nodes. We also observe that in the case of end-to-end sessions bounded within the same IPv4 CSS, the path-info field is not necessary, as it would be composed only of the IP address of the end-node, already contained in the IP header.

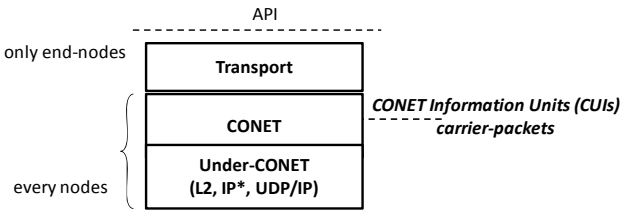

Fig. 3 - Protocol stack

2.3 CONET protocol stack

As shown in Fig. 3, in every CONET node we can find the CONET and the *Under-CONET* layers. The CONET layer is connectionless, handles CIUs and carriers-packets, and provides other functionality (e.g. caching, security, etc.).
The end-nodes has also transport-level functionality, supporting reliability and flow control, and providing the application programming interface (API), see D3.1 [7] for the definition of the API between CONET and upper layers. We adopt the receiver-driven TCP-like approach proposed in [3], which we briefly recall in the following, adapting it to our terminology. The transport algorithm issues a sequence of interest CIUs and each of them requests only a small part of a named-data CIU, e.g. 1500 bytes per interest CIU. By controlling the sending rate of these interest CIUs, it is possible to obtain a TCP-like flow control mechanism. For instance, we could replace current TCP ACKs with interest CIUs and apply TCP congestion-window concepts to in-flight interest CIUs.
Fig. 2 shows the packetization process, the CONET CIUs (interest and named-data) and carrier-packets. We started from the names and structures proposed in [3] and introduced some modifications both in notation and in functionality. As for the notation, the "interest packets" and "data packets" proposed in [3] correspond to our interest CIU and named-data CIU, respectively, but their protocol information is different (e.g. segment info). In addition, we introduce the concept of carrier-packets, with the goal of improving the forwarding speed of CONET.

[1] This info will be used to find the reverse-path to route the named-data CIU back to the requesting node, in the delivery phase. In [3] the same goal is achieved by maintaining states in network nodes. We propose to use source-routing, being aware of the involved trade-offs, and given that we think that the number of traversed CONET border nodes should be rather limited (e.g. CSSs should coincide with Internet Autonomous Systems).
As an alternative, the path-info field could contain the NID of a named-sap, specifying where the requesting end-node can be reached, and the reverse-path routing could be performed by means of route-by-name procedures. This alternative would be more convenient if CSSs are smaller and the number of traversed CONET border-nodes is larger. Also, this alternative would give to the network operator more freedom in choosing the reverse-path.

A named-data (i.e. a content) is split in different chunks. The optimal chunk size is the result of several tradeoffs; we favor a size roughly equivalent to the size of chunks in current P2P systems, e.g. 256-512 kbytes. However, the CONET architecture can support variable chunk sizes.

Each chunk is inserted in a named-data CIU. Named-data CIUs are the data-units of the caching process and their control information include the network-identifier, the chunk number, and temporal and security data.

The network-identifier is a tuple <*namespace ID*, *name*>. The *namespace ID* determines the format of the *name* field. Thus, the *name* field is a namespace-specific string. Each namespace follows its own rules to release unique *names* with its own format. We specified a default naming format, where the *name* is the composition of two hash values, i.e. *name*=<*hash (Principal)*, *hash (Label)*>. Principal and label [2] are flat-names and a hash function transforms them to a fixed number of bytes (e.g., 6 bytes). A principal is the owner of her named-data and uses the *Principal* identifier whose hash is unique in its namespace. *Label* is an identifier chosen by the principal to uniquely differentiate her named-data. For instance, to support the WEB resources we could define the namespace "www", which follows the actual domain name assignment rules and uses the domain name (e.g. www.cnn.com) as principal identifier and the URL path (e.g. /foo/index.html) as label.

The temporal-data include time information, like the expiry date, which can be exploited to implement digital forgetting mechanisms. Security-data [4] make it possible to validate a named-data CIU before caching it or delivering it.

An interest CIU is a request of a set of bytes of a named-data CIU, e.g., from byte X to byte Y (segment info field) of the named-data CIU n. Z (chunk number field).

Carrier-packets are low-level carriers of CIUs and are the data-units of the forwarding process. Carrier-packets are reassembled in border-nodes or in internal-nodes that want to cache the related named-data CIU, and in end-nodes; this operation is necessary to validate the content.

A carrier-packet has the following structure (Fig. 2): i) a *header* field, which transports a minimal set of control information of the CIUs, i.e. network-identifier, chunk number and CIU type (e.g. interest or named-data); ii) a *payload-header*, which identifies the byte boundaries of the carried segment (segment info); iii) the *payload* (existing only in the case of named-data CIU), which contains a part of the sequence of bytes contained in the temporal/security-data and data-chunk fields of a named-data CIU; iv) the *path-info* field, previously described in Section 2.2.

We introduced carrier-packets because a named-data CIU could be too large to be transported by a single under-CONET data-unit (e.g. 1.5kB for Ethernet and 64kB for IP) and thus it requires to be segmented. Moreover, carrier-packets make it possible to perform source-routing; indeed they are strictly related to a specific communication session between an end-node and a serving-node (or a cache).

3. NAME-BASED ROUTING: LOOKUP-AND-CACHE

The name-based routing is the mechanism used to update CONET name-based routing tables, which are used by end-nodes or border-nodes to route-by-name interest CIUs. An entry of the name-based routing table contains the tuple <network-identifier, mask, next-hop, output-interface>; it is like an IP routing table entry, but instead of net-prefixes we have *name-prefixes*, i.e. couples <network-identifier, mask>. Next-hop is the *CSS address* of the next border-node toward the serving-node, as outlined in Sec. 2.2.

In [1][3] the authors suggest to use traditional routing protocols, e.g. BGP or OSPF, to disseminate name-prefixes. We name these approaches *prefix-dissemination*.

We argue that prefix-dissemination could produce big name-based routing tables, because the aggregation of names (i.e., network-identifiers) is not effective, when names do not include information about "where" is the serving node [14]. For instance, if we want to support DNS domain names (as we do), a possible location-based aggregation could be done on the basis of top level domains [3]. However, in the case of generic top level domains (.com, .net, etc.) this would not be effective, as current names are geographically very spread (and numerous: .com names are currently about 90 millions). We also analyzed the .it country-code top level domain and found out that about 30% of .it names are outside Italy, which means that the aggregation would not be very effective also in this case. To support the cases in which name-prefix aggregation is not effective, and since it is does not seem feasible to include all possible names in the routing table, we propose a name-based routing, which we name *lookup-and-cache*. In this approach, a CONET node (end-node or border-node) uses a fixed number of rows of its name-based routing table as a *route cache*. When a node misses the routing info required to route-by-name an interest CIU, it looks up its routing entry in a *name-system* (DNS like) and inserts this entry in the route cache. When all rows are filled in, new routing entries may substitute old ones according to a suitable policy. From a logical point of view, a name-system serves a single CSS and a specific namespace.

If a serving-node is inside the same CSS of the node requesting the routing info, the name-system returns the CSS-address of the serving-node. If the serving-node is outside that CSS, the name-system returns the CSS-address of the egress border-node. If there are more than one serving-node, or egress border-node (due to replication operations), the name-system selects the most convenient destination (e.g. according to known techniques [10]).

Prefix-dissemination and lookup-and-cache approaches can work separately or they can be combined, e.g. by using prefix-dissemination for the most popular named-data and lookup-and-cache for the remaining ones.

4. INTEGRATING CONET IN IP

In this section, we describe a technique to support the CONET in a CSS that is an IP network (IP-CSS), e.g. the CSS n.2 of Fig. 1. The IP network can correspond to the whole public Internet; therefore this technique is a way to offer CONET services in the Internet. We propose a so-called *integration approach*, which: i) does not imply to give up IP, as in the clean-state approach; ii) performs better than a CONET placed on top of IP, as in the overlay approach. The idea of the integration approach is to *make IP itself content-aware*, as follows. We propose to transport the header of a CONET carrier-packet in a novel IPv4 option (or IPv6 extension header), which we name CONET option (see Fig. 2 and [12]). Border and internal CONET nodes of an IP-CSS are nothing else than IP routers extended with CONET functionality.

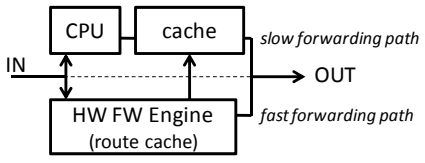

Fig. 4 – Architecture of a CONET node of an IP-CSS

Fig. 4 shows a possible architecture of a border or internal CONET node. We have a *fast forwarding path* that handles forwarding operations for CONET carrier-packets and for plain IP packets. The hardware (RIB or FIB) routing table includes not only IP net-prefixes but also name-prefixes, which address both remote named-data and local cached named-data. The latter entries point to the local cache engine. Other CONET and IP functions with less stringent delay constraints are performed by a CPU. For instance, the CPU performs IP and name-based routing, implements caching algorithms, reassembles named-data CIU to cache them, replies to interest CIUs that request a cached data, etc. Most of these operations require parsing incoming CONET CIUs, which are "copied" in the CPU while at the same time being forwarded by the HW engine.

The advantages of this approach with respect to the overlay one is that it allows CONET nodes to quickly forward carrier-packets, without the need of a slow deep packet inspection. This is a fundamental requirement to deploy content-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 performing 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 of the integration approach is that we require a new IP option, but this is much less disruptive than the clean-state approach. The integration approach lends itself to different deployment scenarios.

It is possible to think to an extreme case in which an IP-CSS corresponds to the whole Internet and routing-by-name functions are performed only in end-nodes. In-network caching would still be possible simply by introducing the new IP option and without the need of introducing routing-by-name functions within the routers.

Another scenario is to partition the Internet in a set of IPv4 CSSs that interoperate only by using CONET protocols. Each CSS uses an IPv4 addressing that is unique only inside that CSS and bounds the scope of IP routing to that CSS. Therefore, new providers offering public CONET services can deploy their networks without necessarily having the availability of public IP addresses, and without increasing the size of the routing table of Internet backbone routers. This scenario could be a solution to the problem of the growing size of Internet backbone routing table [14], moving that problem to the issue of scalability of the CONET routing-by-name mechanisms, which in any case needs to be addressed in CCN architectures.

5. PERFORMANCE CHECKS

This section describes two experimental performance checks.

5.1 Lookup-and-cache

We remind that the routing-by-name process involves only interest-CIUs, since data-CIUs are routed back to the end-node by means of source-routing (see Section 2.2). The CONET nodes involved in routing-by-name are either end-nodes or border-nodes. In case of end-nodes, the lookup-and-cache approach resembles the interaction between an Internet host and a DNS server, where the host implements a local DNS cache service. Therefore, we argue that lookup-and-cache is feasible on end-nodes and we focus on its feasibility in border-nodes.

We assume to replace a standard TCP session between a client and a WEB server with a CONET session (exchange of CONET CIUs) between an end-node and a serving-node, or an intermediate cache. Specifically, we assume that:

- an URL <http://IP address:80 (or domain-name)/path> is replaced by the network-identifier: namespace="www", principal="IP address:80", label="path";
- TCP segments are replaced by carrier-packets that convey segments of named-data CIUs;
- TCP ACKs are replaced by carrier-packets that convey interest CIUs (see Sec. 2.3).

With these assumptions, we can map a real Internet trace, formed by TCP segments and ACKs, to a "CONET trace", formed by carrier-packets. We applied this re-mapping to two Internet traces: the first one captured on an interface at 10 Gbit/s of a tier-1 router [15]; the second one captured on an interface at 10 Mbit/s of a tier-3 router [16].

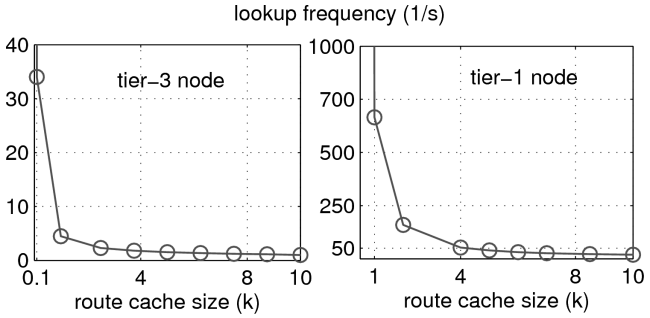

Fig. 5 – Lookup frequency of a tier-3 and a tier-1 border-node

The two re-mapped traces have been fed to a CONET border-node, which we emulated in SW, to analyze the effectiveness of the lookup-and-cache routing for a tier-1 and a tier-3 border-node. Following the approach suggested in [2], we assumed that routing-by-name is performed only on the base of the principal identifier. This means that a name-based routing entry has the form <namespace, hash(principal),*> and that all the named-data of a given principal are stored in a serving-node (and in its replicas, if any). We also assume that the route cache adopts a Least Recently Used (LRU) caching policy, discarding the least recently used item first.

Fig. 5 shows the obtained results in terms of name-lookups per second issued by the border-node to the name-system, versus the size of the route cache. The route caching performance improves (i.e. lower lookup frequency) in a log-like fashion versus the cache size. In the case of the tier-3 node, we have about 2 lookups per second and a cache-miss probability of about 10^{-3}, by using a route cache of 2k entries. In the case of the tier-1 node, we have about 10 lookups per second and a cache-miss probability of about 10^{-4}, by using a route cache of 8k entries. Considering that nowadays BGP routers handle about 350k entries and 2 or 10 lookups per seconds are reasonable values, we can conclude that lookup-and-cache seems feasible with the current technology.

5.2 CONET-IP integration

In this section, we verify the feasibility of conveying the header of carrier-packets in an IPv4 option, i.e. the CONET option. The rationale of this test lies in the fact that IP routers tend to process packets with IP options in the slow forwarding path; therefore, current IP routers could become a critical performance bottleneck for our solution, as plain IP routes and CONET nodes would need to co-exist in a hypothetical real deployment scenario.

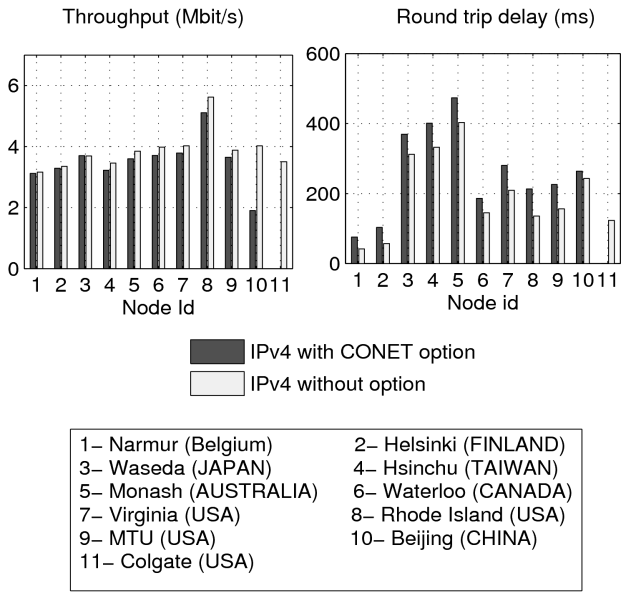

Fig. 6 – Throughput and round-trip-delay of IP packets with and without CONET options on different Internet paths

To check the behavior of current IP routers, we sent IP packets with and without our CONET option (simultaneously) on the on on the Internet and we measured the difference in terms of round-trip-delay and throughput (i.e. the available capacity between a sender and a receiver). We used eleven PlanetLab nodes, spread over the Internet (Asia, Europe, North America, Australia). Each measurement was performed between a PlanetLab node and a node in our premises (Rome, Italy). Each measurement has been repeated ten times and Fig. 6 reports the average values.

As regards the throughput, we observe that we have almost the same performance, with and without the CONET option, for the first nine PlanetLab end-nodes. On the other hand, we observed considerable differences in the case of the last two end-nodes. Further analysis revealed that: i) on the Beijing-Rome path there is a router that statistically drops half of the packets with IP options; ii) on the Colgate-Rome path there is a router (in Australia) that drops all packets with IP options. The problem regards a minority of the examined routers, depends on a software configuration and we conjecture that these policies are enforced to prevent DoS attacks [17]; such policies could be modified, so as to accept CONET carrier packets without restrictions. As regards the round-trip-delay, we observe a small increase of the latency for packets with the CONET option. Overall, our measurements show that IP routers, properly configured, would not be a critical performance bottleneck, and therefore the use of the IP CONET option seems feasible (see also [18] for a similar analysis).

6. CONCLUSIONS

As a conclusion, let us re-consider, in light of our work, the advantages, the cons and the challenges of CCN, which we discussed in the introduction. We argue that our proposed CONET architecture and technical solutions: i) are able to effectively support in-network caching and content replication; ii) support an "integration" approach that can be incrementally deployed in current IP networks; iii) face the scalability limits of name-based routing with the lookup-and-cache approach; iv) do not need to maintain states in network nodes; v) support also communication sessions different from content retrieval, either with the support of named-sap (N.B. this was only mentioned in this paper) or thanks to the fact that CONET can smoothly co-exist with IP networks and therefore such communication session could continue to be run on classical IP.

7. ACKNOWLEDGMENTS

CONET has been devised in the CONVERGENCE project [7], which aims at enhancing the Internet with a content-centric, publish-subscribe service model, based on a common container for any kind of digital data, including representations of people and Real World Objects.

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

[6] 4WARD project website: www.4ward-project.eu

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

[8] K Katsaros, G. Xylomenos, G. C. Polyzos: "MultiCache: An overlay architecture for information-centric networking", Computer Networks, Elsevier, Volume 55, Issue 4, 10 March 2011, Pages 936-947

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

[10] D. C. Verma "Content Distribution Networks", Wiley-Interscience

[11] V. Jacobson, et al "VoCCN: voice-over content-centric networks", ReArch '09 workshop, 2009

[12] A. Detti et al., "An IPv4 Option to support Content Networking", Internet Draft, draft-detti-conet-ip-option-00, Work in progress, March 2011.

[13] D. Oran, "OSI IS-IS intra-domain routing protocol", IETF RFC 1142

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

[15] CAIDA Internet Trace Storage, https://data.caida.org/datasets/passive-2010/equinix-sanjose/20101118/

[16] Waikato Internet Trace Storage, http://www.wand.net.nz/wits/waikato/1/20050815-000000-0.php

[17] F. Gont, S. Fouant, "IP Options Filtering Recommendations", Internet Draft, draft-gont-opsec-ip-options-filtering-00.txt

[18] R. Fonseca, G. Porter, R. Katz, S. Shenker, and I. Stoica, "IP options are not an option", Technical report, EECS Department, University of California, Berkeley, 2005.