Enabling ICN in the Internet Protocol: Analysis and Evaluation of the Hybrid-ICN Architecture

Giovanna Carofiglio
Cisco Systems Inc.
gcarofig@cisco.com

Luca Muscariello
Cisco Systems Inc.
lumuscar@cisco.com

Jordan Augé
Cisco Systems Inc.
augjorda@cisco.com

Michele Papalini
Cisco Systems Inc.
micpapal@cisco.com

Mauro Sardara
Cisco Systems Inc.
msardara@cisco.com

Alberto Compagno
Cisco Systems Inc.
acompagn@cisco.com

ABSTRACT

Information-Centric Networking (ICN) embraces a family of network architectures rethinking Internet communication principles around named-data. After several years of research and the emergence of a few popular proposals, the idea to replace the Internet protocol with data-centric networking remains a subject of debate. ICN advantages have been advocated in the context of 5G networks for the support of highly mobile, multi-access/source and latency-minimal patterns of communications. However, large scale testing and insertion in operational networks are yet to happen, likely due to the lack of a clear incremental deployment strategy. In this paper, we analyze a recent proposal Hybrid-ICN (hICN), an ICN integration inside IP (rather that over/ under/ in place of) that has the ambition to trade-off no ICN architectural principles. By reusing existing packet formats, hICN brings innovation inside the IP stack, requiring minimal software upgrades and guaranteeing transparent interconnection with existing IP networks.

We describe the architecture and use the open source implementation to test hICN in the open Internet to validate its short-term deployability. Further, we consider linear video streaming over mobile wireless heterogeneous networks as use case to highlight hICN advantages compared to TCP/IP counterpart.

CCS CONCEPTS

• **Networks** → **Network architectures**; **Layering**.

KEYWORDS

Future Internet architectures, ICN, IPv6

ACM Reference Format:
Giovanna Carofiglio, Luca Muscariello, Jordan Augé, Michele Papalini, Mauro Sardara, and Alberto Compagno. 2019. Enabling ICN in the Internet Protocol: Analysis and Evaluation of the Hybrid-ICN Architecture. In *ICN '19: Conference on Information-Centric Networking, September 24–26, 2019, Macao, China.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3357150.3357394

ICN '19, September 24–26, 2019, Macao, China

ACM ISBN 978-1-4503-6970-1/19/09…$15.00
https://doi.org/10.1145/3357150.3357394

1 INTRODUCTION

Information-Centric Networking (ICN) identifies a network architecture built upon named data, not host location, for a simplified and more efficient user-to-content communication.

Despite differences in ICN architectures, a common shared idea characterizes them: scalable location-independent communications with data-centric security. This results in a native support for mobility, storage and security as network features, integrated in the architecture by design, rather than as an afterthought. Several years of research and in-lab experimentation have advanced the architectural design and contributed to show ICN potential. However, the ICN idea still divides the community because of the "gain/pain" trade-off [36] related to ICN introduction in existing IP networks.

Recently, a regain of industrial and academic interest in ICN has been generated by the need for network designs capable to face future 5G network challenges. The next generation of radio-mobile networks strives to serve a large number of use cases across several vertical markets and ICN has been identified as one promising candidate to bring the required benefits at the network edge in terms of performance, scalability and cost [2, 48, 66]. 5G architectural discussions have also revived the debate about deployment path and cost for ICN introduction in operational networks.

Even if virtualization and application-centric network slicing in 5G may accommodate the use of new data planes like ICN [73, 75], skepticism remains about short term clean slate ICN insertion. A partial integration of ICN semantics into IP has been looked at in the past to offer an easier introduction in the existing protocol stack at the cost of modified ICN behavior and trade-off of its benefits.

A new recent ICN proposal by Cisco called hICN has been published as Internet Draft [64] and is currently work in progress at the IETF. Moreover, an open source implementation has been made available in the Linux foundation *Fast Data* project (https://fd.io).

The contribution of this paper is twofold: (i) we describe and analyze the hICN network architecture in detail with major emphasis on the trade-offs between features and deployability; (ii) evaluate the performance of the current open source implementation for use cases that have at the same time practical interest and highlight ICN features.

The reminder of the paper is as follows. hICN design is presented in Section 3, showing how the architecture preserves all the core features of ICN (described in Section 2). We expect a deployment strategy for hICN that targets few nodes at the network edge, leveraging the transparent interconnection between hICN and standard

IP routers. A proof of the feasibility of such deployment is presented in Section 4, where we show that hICN traffic can traverse a large fraction of Autonomous Systems in the Internet. Section 5 further illustrates some of hICN potential benefits for live video streaming over mobile and heterogeneous networks, as inherited by ICN, and quantified over traditional IP network and transport layer approaches. Rather than on the novelty of demonstrated ICN advantages, the focus of the assessment is on the ability to fully realize ICN gains at minimum integration cost in the existing IP infrastructure, both from the network and the application point of view. Finally, Section 6 and Section 7 respectively provide a review of related work and a short discussion of next steps.

2 ICN KEY FEATURES

In this section we briefly review what we think to be the set of key ICN compelling features as highlighted in earlier research. The goal is to analyze if these features are preserved in the hICN design. To do so, we focus on CCN/NDN as reference ICN designs. The most updated reference for CCN is the set of two RFCs [62], [63], while for NDN we consider the online project specifications (https://named-data.net/project/specifications/).

▪ **Named Data** – In ICN, information is addressed by location independent identifiers and network operations are bound to named-data, not location. The basic idea is to enrich network-layer functions with content awareness so that routing, forwarding, caching and data transfer operations are performed on topology-independent content names, not on IP addresses. Data are divided into a sequence of packets uniquely identified by a name (called *data* packets) and fetched by the user in a *pull-based* connection-less fashion via named packet requests (called *interests*). Naming data packets allows ICN network to directly interpret and treat content according to its semantics, with no need for DPI (Deep Packet Inspection) or delegation to the application layer.

Even if ICN naming is still an open area of research, a few lessons can be drawn from past research and experimentation [15, 23, 40, 41, 63]: (i) ICN need not to specify a naming convention which can be instead application-specific; (ii) a hierarchical structure is recommended for routing scalability to guarantee entries aggregation in name-based routing tables; (iii) names are not necessarily human-readable but may be hash-based; (iv) scalability of name-based routing in inter-domain use cases and in presence of producer mobility is still an open research area.

▪ **Dynamic Forwarding** – Name-based data plane makes use of soft state [96]: user requests are routed by name and a trail of pending requests is temporarily left in the router to guarantee reverse path forwarding of corresponding data. Additionally, the presence of such pending requests in routers enables request aggregation (and native multicast), dynamic re-routing and application/network aware *forwarding strategies* (e.g. based on popularity, on network status, for multi-path load-balancing, etc.) [78, 97]. The question about feasibility of ICN forwarding pipeline has triggered various studies with promising results in the last years [30, 67, 84, 99].

▪ **Data-Centric Security** – Instead of securing connections, ICN model is based on securing data at network layer. Each data packet is digitally signed by the producer, allowing consumers to verify *integrity* and *data-origin authenticity*. A producer is thus required to have and distribute at least one public key. Existing trust models (*e.g.* a PKI or Web-of-Trust) can be used to validate producer identity and key ownership. *Data confidentiality* can be guaranteed by encrypting data payload and preventing information leakage from the name as proposed in [39].

Beyond the still open research challenges surveyed in [14, 61, 65], one commonly recognized benefit of ICN data-centric security approach is that it places trust in producers rather than in hosts that store and serve data. This enables in-network efficient data delivery operations, such as filtering, caching and multicasting, without affecting the data security properties enforced by the data producer.

▪ **Receiver-Driven Connection-less Transport** – In contrast with the current sender-based TCP/IP model, ICN transport is receiver-controlled, it does not need connection instantiation and it accommodates retrieval from possibly multiple dynamically discovered sources. ICN transport builds upon the flow balance principle, guaranteeing corresponding request-data flows on a hop-by-hop basis [5]. A large body of work has looked into ICN transport (surveyed in [76]), not only to propose rate and congestion control mechanisms [79, 100] – especially in the multi-path case [29] – but also to highlight the interaction with in-network caching [28], the coupling with request routing [29, 47], and the new opportunities provided by in-network hop-by-hop rate/loss/congestion control [32, 93] for a more reactive low latency response of involved network nodes.

▪ **Other features of the ICN architecture** – A result of the above mentioned ICN properties is support for **in-network caching**. The ability to perform a name lookup in router buffers can be exploited for re-use (asynchronous multicast of data via cached replica) and repair (in-network loss control), which is an important differentiator w.r.t. existing end-to-end solutions. Many studies have proven its advantages [33, 77, 101, 102], but also the differences w.r.t. application-level CDN-like caching [31, 95].

Another important implication of ICN naming, security, forwarding and transport model is **native mobility** support. Previous work has highlighted the benefits of ICN seamless mobility, especially in 5G context [19, 37, 51, 74, 89, 103], as mostly deriving from its "anchor-less" management approach in the data plane. As a result, ICN can handle mobility with no need for a stable point of passage of traffic, rendezvous point or name-location mapping system.

3 HICN DESIGN

The main goal of the hICN architecture is to bring ICN capabilities into existing IP networks, while guaranteeing incremental deployment in networks where only few strategic nodes are hICN enabled. More specifically, hICN *integrates ICN in IP* and, unlike other proposals which are described in Section 6, it does not use any encapsulation nor tunneling techniques, and does not run as an overlay network. hICN by design assumes to share the same infrastructures with regular IP traffic.

Three major principles can be identified in the design: (i) do not sacrifice any of the ICN features; (ii) transparently interconnect hICN routers with standard IP routers, that will forward hICN packets as normal ones, as well as hICN routers will forward standard IP packets; (iii) reuse most of the existing software and hardware

technology, in order to minimize the effort to adopt hICN in the near future. In the rest of this section, we describe the hICN design and validate if and how ICN key features, as presented in Section 2, are preserved.

3.1 Named Data

As in ICN, hICN addresses each piece of data by name. In hICN, content names are network level names used by hICN routers to forward packets. We stress that content names are different from application level names (e.g. the URI for the identification of a web object) which are application dependent and hICN routers are oblivious to them. The mapping between the application name and the network name is made by applications and is out of scope for this paper. However, this topic has been partially treated in [81].

An hICN name is made of two parts: the *name prefix* and the *name suffix*. The name prefix is used by routers for the forwarding operations while the suffix contains the segmentation information and it is mostly used for transport purpose. The concatenation of those two components generates unambiguous names, which uniquely identify data. hICN name prefixes are standard IP addresses that are assigned by the network administrator for this specific purpose. In particular, it could be envisaged the creation of a reversed address family. As described later in this section, this can help during the forwarding operations to distinguish between standard IP and hICN packets. Notice however that this is not mandatory: to identify an hICN packet is enough to know the list of prefixes used to route content and they are available in the hICN routers FIB which could be entirely managed by the control plane.

hICN inherits the ICN request/reply protocol semantics [49]: an interest packet is used to request a data packet carrying the actual payload. The definition of the two protocol data units encompasses both network and transport headers. They are standard IPv6 and TCP headers where we modified the semantics of a few fields. The most important fields are the `Name Prefix` and the `Name Suffix`. The former is carried in the IP destination address field for interest packets, whereas it is placed in the IP source address field for data packets. The latter is written in the TCP sequence number field and carries the segmentation information used by the hICN transport layer. The soft-state-based, hop-by-hop forwarding of data packets is implemented by having hICN-capable routers rewrite the IP header source (respectively destination) address of Interests (respectively Data) as they forward packets. A detailed description of the fields carried in the packet headers and forwarding behavior is available in [64].

The modifications presented above allow to carry the information required by the hICN forwarding and transport layer, while preserving protocol layer separation and compatibility with standard IP routers. This will be further analyzed in Section 4 by experimentation in the public Internet. IPv6 is presented as the reference technology for hICN, although the design should be able to run hICN using IPv4 packets too. Of course, the greater addressing space of IPv6 allows for more flexibility in name encoding. The use of the TCP header for hICN provides simpler network traversal as it will be shown in Section 4. Other standard headers may be used like UDP which is instrumental for the deployment in client operating systems where non privileged applications can easily forge UDP packets. The current hICN architecture and implementation already supports UDP which, together with IP, may constitute in the future the new Internet waist on top of which innovative transport protocols can be built, notable examples are LEDBAT (e.g. BitTorrent and OS upgrades, [82]) and Google QUIC (e.g. Chromium [53]).

3.2 Name management

End hosts must provision additional IPv6 prefixes (at least one /128) to produce named data. Provisioning several IPv6 prefixes is a standard operation that does not need any new special mechanism. The hICN host stack is responsible for provisioning prefixes that will be used as names by applications running in the host. For clarity, we can assume that a host requests name prefixes by sending a provisioning request to a network service similar to standard DHCPv6, or an extension of it. Such a network service is the actual owner of the prefixes that are temporarily leased to the host. Similar to what happens today for interface identifiers, the lease can be static or dynamic and may require the host to authenticate (e.g. 802.1X). The actual owner of the prefix is the entity that guarantees that prefixes are routable in a given domain, private or public. This does not constitute any difference with what happens today in IPv6 address space management. In hICN a host does not announce prefixes to the network, it is the local autonomous system to announce routable name prefixes to neighbors. The latter may look like a tautology as current network management works exactly in this same way. This means that hICN name prefix management inherits all protocols and mechanisms currently used for interface identifiers (IPv6 addresses). The consequence of that is that routing over name prefixes can reuse all routing protocols currently used in the Internet. The ability of today routing protocols to provide several routes to reach a give prefix is significant, even though poorly used mostly to avoid traffic instabilities and out-of-order delivery that IP and TCP cannot manage well. These two problems are addressed well in hICN thanks to local flow balance which provides a simple way to make traffic engineering dynamic, yet stable. The consumer end point is also responsible for taking care of out of order delivery.

As clarified above, in hICN it is not the application that owns name prefixes, but it is the host that leases them from a network administrator. In this respect, hICN may resemble to protocols such as ILNP [16], LISP [35] where the host provision an hostname or a host identifier (ILNP or LISP) from a network service.

3.3 Dynamic Forwarding

The data structures implemented inside an hICN forwarder are similar to those required in an ICN one. A major difference between hICN and ICN forwarders is that the former can reuse some of the existing IP data structures. An example is the Forwarding Information Base (FIB), which in hICN is in fact a regular IP FIB, where hICN name prefixes coexists with normal IP addresses. The FIB can be populated with hICN names using standard IP routing protocols to distribute them in the network (such as ISIS, OSPF and BGP). However, multi-path and multi-source are two important properties that routing algorithms should provide in order to support hICN forwarding strategies. Unfortunately, multi-path and multi-source are poorly supported by current IP routing. Approaches leveraging BGP for anycast routing can be instrumental to obtain multiple

routes to program forwarding strategies, with no guarantees of optimality though. Recent work in the field has shown this to be feasible in an efficient way [38]. An ICN forwarder also contains the Content Store (CS) and the Pending Interest Table (PIT): the former stores the data packets received by the router to reuse them for future requests; the latter keeps track of the forwarded interests to route the corresponding data packets on their reverse path. These two data structures are required in hICN forwarders. In order to minimize the modification required to a standard IP router the CS and the PIT can be merged in a single data structure, called *packet cache*, that can be used to store both kind of packets, with different insertion/eviction policies. The packet cache is indexed by full name and is implemented exploiting the memory buffers already available in the IP routers; it maintains the reverse path information needed for forwarding Data packets (see [64] for details).

3.4 Data-Centric Security

hICN inherits ICN data-centric security model: integrity, data-origin authenticity and confidentiality are tied to the content rather than to the channel. In particular it is possible to provide integrity and data-origin authenticity in two different ways: (i) using an authentication header or (ii) a transport manifest.

The authentication header carries the signature of the data packet and some information about the original producer. The signature is computed over the immutable fields of the data packet, while the others, including the signature are set to zero. This header is added at the beginning of the data packet payload, and is meaningful only to hICN-enabled routers. In this situation, a bit in the IP/TCP header is set to one.

The transport manifest, designed for ICN in [25], is a L4 entity generated by the producer which contains the list of names in a group of data packets. Each name is associated to a cryptographic hash computed over the corresponding data packet. A client has to request a manifest to the producer to know the available data packets and to verify them. Data packets carrying a manifest have the `MAN` flag set to one. Using this method, the producer needs to sign only the manifest packets, minimizing the overhead due to packet signature. This approach in fact guarantees a level of security equivalent to individual packet signatures. Not every application can take advantage of the manifest, such as voice over IP.

The ICN data-centric security model mandates that the *linkage* between name and data be authenticated in order to guarantee secure location-independent content retrieval [83]. NDN or CCNx, create a secure bind between the name of the data, the data itself and the producer identity. Therefore, a consumer can verify if the data is signed with a trusted key and can validate the signature to check authenticity of the data with respect to the name carried in the packet. In hICN this mechanism is left unchanged as the signature covers both network name and content payload.

The mapping between application and network names must honor this security feature. Signature verification validates the linkage between the hICN (network) name and the data, but it does not give any guarantee about the linkage between the application name and the data. If the mapping between application and network names is not verifiable by a consumer, this might expose the hICN architecture to an attack in which the consumer requires data with an application name A, but it is actually translated into a network name that correspond to an application name B. For the attack to work, data B must be signed with the same trusted key expected for content A such that the linkage between the network name, the data and the producer is respected. In order to prevent this attack, the mapping between the application names to the network names must be an injective function defined by the producer and validated by the consumer. One way to achieve such secure mapping is to exploit a global name resolution service, such as GNRS [59]. However, deploying a new global system is not an easy task and it might prevent a simple deployment of hICN. A second approach is to exploit the record Address Prefix List (APL) [52] of the DNSSec system in order to map an application prefix to a hICN name prefix. Once a prefix is mapped, each application can define its own mapping function to further map an application name to the obtained hICN name prefix. A third approach consists in letting applications exploit their own mapping system as in the case of applications that use the Session Initiation Protocol.

Trust management is also similar and hICN can benefit from the most recent work in ICN research [98] on the topic. Additionally, the network service described in Section 3.2 serves to bootstrap trust between applications acting as a Certification Authority. When requesting a network prefix, the producer will send its public key to the network service as well as his identity. The network service will create and sign with his private key an hICN data including the producer's identity, the producer's public key, the prefix assigned to the producer, and the time of lease. The hICN name of such data is assigned by the network service and used as key locator for the producer's public key. Trust on the network service can be achieved with any existing trust model (e.g., PKI or Web of Trust), or exploit existing trust systems already deployed (e.g., the BGPSec trust model where the private and public key are those corresponding to the RIR in the RPKI).

For what concerns confidentiality, this is delegated to an upper layer secure transport that we do not further discuss in this paper, ICN and hICN do not differ on this respect. More research is needed in this area and left for future work.

3.5 Receiver-Driven Connection-less Transport

Transport in hICN is similar to ICN: it is *receiver-driven*, *connection-less* and supports *multiple point*. All these features allow for in-network caching, in-network loss and congestion control as well as bandwidth aggregation over heterogeneous networks. The current hICN software uses RAAQM, the receiver-driven congestion control proposed for ICN [29]. The protocol discovers and exploits available content sources in the network, maximizing the bandwidth available at the consumer. The implementation also includes wireless optimizations such as in-network loss control mechanisms introduced in [32]. Both protocols are used in the evaluation, in Section 5.

In addition to congestion and flow control, the ICN transport layer, and so the hICN one, needs to provide data authentication and data integrity verification, as discussed in the previous section. These features are all implemented inside the hICN sockets. Similarly to standard sockets, there are different options that the application can specify for an hICN socket. Some options are the

same as standard sockets (select between stream or datagram oriented flow, reliable or unreliable transfers), others are specific to hICN. An hICN socket can be a consumer or a producer socket and as opposed to standard INET sockets, hICN sockets are unidirectional: the producer socket generates the data while the client side request them. An application may need to act both as consumer and producer at the same time: in this case it simply opens two different sockets. In addition, applications specify how to provide integrity and data-origin authentication: signing packet by packet (using the authentication header) or by using the transport manifest. Once the producer socket has processed a block of data coming from an application, this is stored in a portion of memory managed directly by the underlying forwarder. The size of this memory portion can be decided by the application using another socket option. More details about hICN transport design and implementation can be found in [81].

3.6 Other features of the hICN architecture

As with ICN, the design of hICN allows for in-network caching and native mobility support. We already discuss about caching in Section 3.3: each hICN forwarder is equipped with a packet cache that stores data packets which can be reused to satisfy future requests. Consumer mobility is fully supported by hICN thanks to name-based addressing. Producer mobility instead requires specific management protocols as in ICN [103]. In particular, the current software implementation provides the anchor-less Map-Me micro-mobility service as described in [19]. Additional work in progress on this topic is done in the IETF DMM WG [20, 21] that we do not analyze in this paper.

4 FEASIBILITY ASSESSMENT

We report results of an experimental campaign of Internet measurements performed to observe hICN traffic in existing IP networks. End-to-end reachability, middlebox traversal and compatibility with regular IP routers are the target of our evaluation to prove feasibility of hICN insertion in an increasingly "ossified" Internet architecture [45, 46]. In our experiments, we collect empirical evidence that semantic changes in IP/TCP header fields, as well as the lack of underlying TCP state machine, does not result in intermediate nodes dropping, corrupting or interfering with hICN traffic. The optional TCP pseudo-header over UDP does not require any further reality check. In this section, we report our observations related to IPv6 measurements. Our results however apply to the IPv4 context, despite the large number of encountered NATs and connection trackers. A more detailed report is left for future work.

4.1 Controlled End-to-End Deployments

We enable hICN in a set of representative nodes in academic, residential, enterprise and cloud environments, and transfer content from an hICN-enabled producer to an hICN-enabled consumer over an IP only path, using the producer's IP address as unique content name. The aim of these tests is to validate the open source implementation and tune hICN in order to traverse the most common types of middleboxes.

All the tests conducted were successful, meaning that the consumer nodes were able to retrieve the required content. However, we found devices that were interfering with hICN traffic. Table 1 summarizes the types of devices found during the tests, and the counter-measures we put in place. Stateful firewalls can be traversed by setting well-known destination ports, as well as setting the `SYN` flag on interests, and consequently `RST/ACK` flags on data packets. We decided to use the `RST` to not overwhelm connection trackers. Enterprise contexts are the most problematic, as security appliances can for instance mangle TCP sequence numbers, altering the name of the hICN packets. In these scenarios we have no choice but to use a tunnel. We implemented these features as new face types in the forwarder, so that they can be selectively applied in a hop-by-hop fashion, thanks to the connection-less nature of hICN. We also noticed during the tests that devices performing deep or stateful inspection of traffic are most likely situated close to endpoints. This means that we can limit the overhead introduced by these faces at first hICN-hop only.

Context	Issues	Counter-measures
Academic	None	None
DC/Cloud	None	None
Residential	Stateful firewall	SYN for Interest, RST/ACK for Data
Enterprise	Security appliance	First-hop tunnel

Table 1: Summary of end-to-end hICN measurements.

4.2 Large Scale Measurements

The tests in the previous section have the intrinsic limitations of the end-to-end approaches [22] and they have been conducted on a small number of controlled nodes. To scale up our test we conduct a traceroute-like test where we send TTL-limited probes towards every announced IP prefixes. Upon expiration, those packets eventually elicit an ICMP response from the routers on path. The ICMP response contains a copy of the original headers and it both acts as a proof that the packet made it up to that point, and also reveals alterations, if any. We perform our test using hICN packet headers populated with values characteristic of interest and data packets and using different variations on TCP flags.

To run our test, we extract IP prefixes from Routeviews datasets [9]. The dataset contains 48619 prefixes announced by 14398 ASes. According to the CIDR IPv6 report [1] there are 14455 ASes in the routing system at the time of writing so our dataset covers almost the entire Internet. To map each IP address discovered during the tests to its AS, we use the Team-Cymru IP-to-ASN service [13]. We further annotate and categorize the ASes based on PeeringDB information [4].

The procedure adopted in our test is illustrated in Figure 1. In the first phase, for each prefix in our data set, we select a representative IP address (taking the first address in the subnet) and we send traceroute-like traffic with increasing TTL toward it. We keep increasing the TTL until our probes enter the destination prefix or they cannot go further. This allows us to map the set of responsive hops at a given distance (we account for load balancers by using paris traceroute [17]), and to verify the absence of rate limiting by sending small packet bursts. In the second phase we send

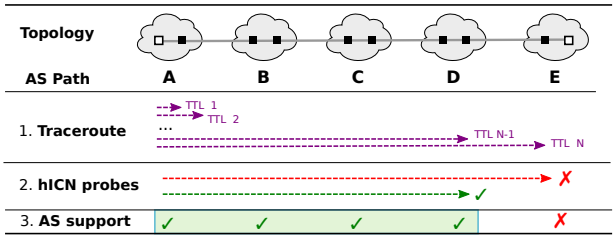

Figure 1: Illustration of one radar-like measurement to validate hICN packet support by intermediate AS.

TTL-limited hICN probes to the responsive hops by proceeding backwards from the stopping point of the previous phase. We verify the reply TTL, and log whether hICN traffic is accepted, dropped, or altered by intermediate nodes. We consider a successful reply to our hICN probe an ICMPv6 packet of type 3 (time exceeded) and code 0 (hop limit exceeded in transit). In addition, the header in the payload of the reply must be the same as the header of the hICN probe, except for its hop limit counter. A successful response terminates the measurements and mark all previous hops as hICN compliant. Finally, IP-to-AS mapping uncovers the underlying topology and help us to infer AS-level metrics, taking into account border effects resulting from uncertainties at the AS border.

	Total	Coverage		hICN support	
	#AS	#AS	ratio %	#AS	ratio %
Cable/DSL/ISP	2291	1169	51.03	1032	88.28
Content	914	480	52.52	423	88.13
Academic	290	181	62.41	160	88.40
Enterprise	257	90	35.02	81	90.00
Non-Profit	211	103	48.8	85	82.52
Not Disclosed	850	355	41.76	309	87.04
Service Provider	1483	991	66.82	912	92.03
Route Server	18	10	55.5	10	100.00
Unknown	8104	2530	31.21	2155	85.17
TOTAL	14418	5909	40.98	5166	87.4

Table 2: AS-level support of hICN as revealed by our large scale measurements.

We report the results of our tests, aggregated at AS-level, in Table 2. Despite its simplicity, our approach manages to sample a representative subset of the overall IPv6-enabled ASes, both in number and diversity. Analysis of these data in light of AS-level topologies provided in [11] revealed that most missing AS are stubs without customers, confirming that we are indeed well-covering Internet core. We noticed that our approach reports missing measurements when approaching the targeted destination: in 95% of the cases this happens when we reach a small AS and suggests filtering of traceroute traffic. Finally, we observed several cases where the IP addresses that we used belonged to a non-routed prefix. This is mostly due to prevalent prefix aggregation close to destinations. In the future, the usage of a hit-list will allow us to increase the AS coverage, especially of stub and small ASes.

The results obtained also confirm our findings in the end-to-end experiments: most of the middleboxes that may interfere with hICN traffic are deployed at the edge. In fact, we tested all the countermeasures we identified in the previous section, but results reveal only negligible differences with respect to plain hICN packets. This is due to the fact that we mostly covered the core of the Internet, where there is a small chance to hit a middlebox.

As a last analysis, we consider the ASes that are carrying more than 1Tb/s of traffic, which are 138 according to PeeringDB. The results show that hICN is able to traverse all of them. We remark, in the end, that our results only provide a lower bound of success, since we cannot conclude if an AS supports hICN or not in absence of response to our probes.

Overall, these results are promising and confirm our suggestion to deal with problematic equipment close to the edge through specific faces, leveraging the hop-by-hop capabilities of hICN. Core routers and servers can then only be limited to the plain version of the protocol for minimal state and maximum processing performance.

5 LINEAR VIDEO DISTRIBUTION

The goal of this section is to verify that the current design and implementation meet all the initial design principles: (i) hICN does not trade-off any of the ICN features, (ii) it allows for interoperability between hICN and plain IP nodes and (iii) it can be incrementally deployed, enabling hICN only on few selected nodes. We consider Adaptive Bit Rate (ABR) linear video distribution use case to show the benefits of hICN. The open source hICN software distribution provides ABR application support, that we use in this set of experiments. Linear ABR video distribution is a challenging use case in general as it requires provable QoE in terms of video quality, application responsiveness and it is also supposed to scale to a very large number of watchers. Serving millions of concurrent video streams with the usual broadcast TV quality is a challenge faced by all big players in content distribution, often via proprietary in-house CDN-like solutions [54]. These technologies are at an early stage, providing limited support for rate adaptation and mobility. ABR video streaming for linear video is a use case which has already triggered attention in the ICN community and different works have shown several advantages in using this technology [26, 42, 55, 56, 60, 68, 70, 72, 80, 94].

5.1 Workload and Implementation

In our tests the content source consists in a live video feed sent by the Open Broadcaster Software (OBS) [3] sending a RTMP stream to a *nginx* [6] server providing multi-quality HLS streams through the `nginx-rtmp` [7] module. We stream 48 channels, each one encoded in 4 qualities (using bit rates suggested in [10]) with 2 seconds segments, ranging from 360p at 1Mbit/s to 1080p at 6Mbit/s.

The full hICN stack is based on the Linux Foundation open source project *Fast Data* at https://github.com/FDio/hicn, https://github.com/icn-team and https://hub.docker.com/r/icnteam/.

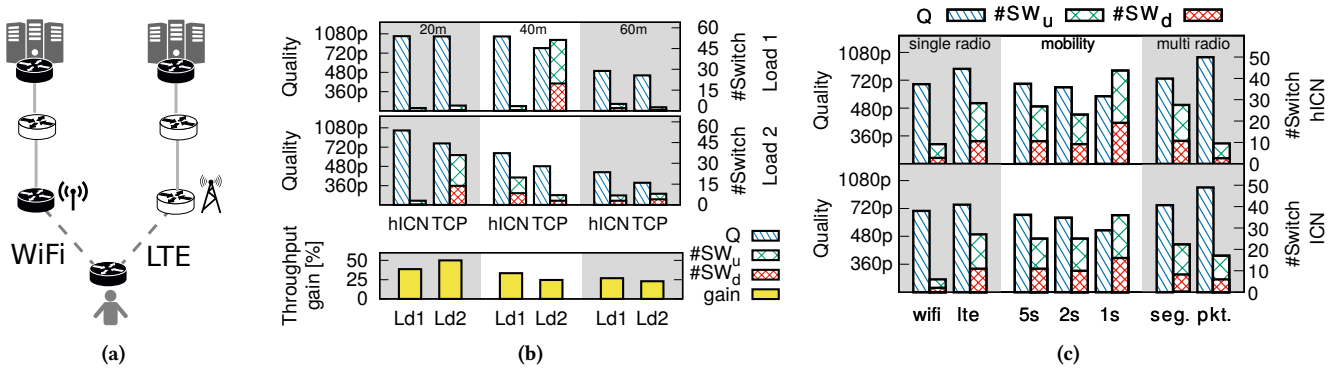

Figure 2: (a) Video distribution over HetNet. Black routers are hICN enabled. (b) TCP vs hICN over WiFi. Average video quality retrieved by the client (Q) and number of quality switches up ($\#SW_u$) and down ($\#SW_d$) with different cross-traffic load on the WiFi channel. The bottom plot shows the throughput gain of the hICN flow with respect to TCP. (c) Hetnet Access. Comparison between hICN and ICN in case of mobile client and bandwidth aggregation.

At the server side, we use a forwarder based on the high-performance Vector Packet Processing framework (VPP) [58]. This implementation consists of a plugin that adds hICN-specific processing nodes to VPP. Initial benchmarks with a point-to-point workload and realistic mixed packet sizes show that this prototype can easily saturate a 10Gb/s link using a single worker thread. At the client side, the forwarder is implemented as an user-space library (hicn-light). The ABR video HTTP cache is based on the Apache Traffic Server which uses hICN through a plugin which is also available in the Fast Data project. This forwarder achieves about 400Mb/s throughput with a single thread and runs on all major operating systems. For our experiments we use Ubuntu Linux clients using the VIPER video player, also available in the Fast Data project. This player provides different adaptation logic strategies for ABR video and, in our experiments, we use the ADAPTECH strategy [80]. The player is able to retrieve content using the default IP/TCP stack, as well as the ICN and hICN ones. In ICN/hICN mode, the end-points use RAAQM [29], a receiver-driven multi-path congestion control that allows to use multiple paths. For better parameter control, all radios are based on realistic emulation capturing effects of distance, path loss and fading. Details of the emulator can be found in [18].

5.2 In-network Control

A compelling feature of ICN transport is that it enables efficient in-network rate/loss/congestion control operations [27, 32, 93], resulting from the combination of pull-based request, symmetric hop-by-hop forwarding and in-network caching.

In this section, we consider the network in Figure 2a where a client is connected to WiFi only. hICN is enabled both at the user and server size, and also in the Access Point (AP), leaving a regular IP router between the AP and the server. By enabling hICN in the AP, we can benefit from the Wireless Loss Detection and Recovery (WLDR) algorithm introduced in [32] which is available in the open source implementation. Here we show that enabling hICN only on few nodes we can get the same advantages that we could have gained using a full ICN network with respect to a standard TCP transport.

Figure 2b compares the performance of one CUBIC TCP and one hICN flow over WiFi with the user at 20, 40, 60 meters from the AP, as indicated in the top part of the plot. During the experiment we generate UDP cross-traffic on the wireless channel using MGEN [12] that accounts for average loads of 25% and 75% of the available bandwidth (resp. *Load 1* and *Load 2* in the plots). We perform 5 rounds, during which the client watches 5 minutes of a single live channel. The charts report the average values over all runs. The top part of the figure reports the average video quality Q downloaded by the client and the number of video quality switches, to a higher (switch up) and lower (switch down) quality, respectively $\#SW_u$ and $\#SW_d$.

Our tests demonstrates that hICN gets a better average video quality with respect to TCP and, most of the time, less quality switches, which means an improved QoE for the client. This is thanks to WLDR that recovers losses locally, instead than end-to-end as in TCP. The higher number of switches with hICN at 40m, 75% load can be ascribed to a limitation of the adaptation logic we use. We can confirm this by measuring the average throughput measured by the application, which we plot as the relative gain of hICN over TCP at the bottom of Figure 2b. In this smaller plot Load 1 and Load 2 are indicated with $Ld1$ and $Ld2$ respectively. hICN gets consistent superior performance with respect to TCP, ranging from 23% (60m, 75% load) to 50% (20m, 75% load) gain, demonstrating that we can achieve the same benefits of ICN deploying hICN only on few nodes.

5.3 Seamless Mobility

In this section, we consider the HetNet access scenario in Figure 2a. In this scenario the user has access to WiFi and LTE radio access technologies. The two radios are connected, through different networks, to two distinct live feeds, providing the same video channels. Both radios in the experiments use realistic emulation. As for the previous test, we perform 5 runs, during which the user watches 5 minutes of a single channel. In the following we compare hICN with ICN, to highlight that the two network architectures are actually equivalent and they bring the same benefits. To run the ICN

scenario we enabled ICN on all the nodes (using the CICN implementation [57]), while, for hICN, we enabled again only the user, the AP and the two servers. WLDR is active on the WiFi channel both for ICN and hICN.

We start with the two baseline behaviors over WiFi or LTE only, without any mobility event. The results are displayed on the left and labeled *single radio*. As for the previous test we show the average quality and the number of quality switches. The results for the two architectures are comparable. For the WiFi channel we can conclude that WLDR works in the same way both in ICN and hICN. Using only LTE the client gets a higher average quality at the cost of more switches, and again we can attribute this instability to the application adaptation strategy. The middle plots, labeled *mobility*, show the same metrics when the consumer performs handovers between WiFi and LTE every 5, 2 and 1 seconds respectively. Again, the results are almost the same for the two architectures. In particular, thanks to the native consumer mobility support of ICN/hICN, the video quality is not highly affected, despite the fast mobility of the client. Being in control of the request process, the consumer can adapt its interest sending rate according to measured available bandwidth on each path, eventually accounting for newly available interfaces. The performance shown in the chart might be further improved by supporting the client-side mobility recovery mechanisms described in [32], though not available in the current implemenation.

5.4 Bandwidth Aggregation over HetNet

The receiver-driven transport property of ICN/hICN permits a simple but efficient realization of channel bonding over heterogeneous radios, through the use of congestion aware load-balancing, implemented as a forwarding strategy [29]. It acts at packet level with the objective of minimizing the residual latency on every available path. This scheme can furthermore be applied network-wide and brings full multi-homing/multi-path/multi-source support. In this experiment, we use the same setting as in the previous section, showing that the performance of ICN and hICN are consistent. We test two different forwarding strategies that implement two different load-balancers: a per-dash-segment load-balancer, which tries to mimic the granularity available for applications today, and the per-packet load-balancer just described. Results are in the right-hand side of Figure 2c, labeled *multi radio*. The load balancing at segment level (labeled *seg.*) leads most of the time to performance degradation, which is consistent with different observations made in previous work on MPTCP [50], suggesting that those technologies are more appropriate for fast recovery than for channel aggregation in the case of DASH video streaming applications. Per-packet load balancing (labeled *pkt.*) instead achieves an optimal traffic split over the two channels, fully exploiting available bandwidth without any incidence on the application, despite the intrinsic differences between radios. By bonding WiFi and LTE, our client obtains the highest quality with a minimum number switches. Also in this case, both the hICN and ICN implementation are comparable.

We want also to highlight that our client downloads from two different sources at the same time, using disjoint paths and no proxy at junction points. To the best of our knowledge there is no transport protocol nowadays that is able to do the same. This is another nice property of ICN inherited by hICN.

5.5 Scalability

In this section, we show two key benefits of hICN for linear video distribution at scale: 1) using hICN, the server load scales with the number of channels, rather than with the number of connected users as in TCP/IP; 2) hICN deployment at the network edge in addition to endpoints, yields to traffic offload not only at the server, but also in IP network core. We consider the topology in Figure 3a, where a cluster of video clients is connected to a video server. Every client is connected through a 50Mb/s link to the edge routers, while the other links in the topology are 10Gb/s links. The video server is an Apache Traffic Server (ATS) [8], configured as an HTTP reverse proxy with a 2GB cache (1GB of RAM cache and 1GB of raw device cache). We implemented a plug-in that uses hICN sockets to interface ATS with the hICN forwarder. Every hICN forwarder has 750MB packet cache size. We run 2 hours experiments with TCP/IP and with hICN under different client population (specifically 100, 200 or 300 clients). Each client requests one of the 48 available channels (encoded in a single video quality) and switches to a different channel every 10 minutes. Channel selection follows a Zipf distribution with $\alpha = 1.4$.

In the experiments, we considered three different scenarios: (i) called *TCP*, in which every router and endpoint uses TCP/IP; (ii) called *hICN endpoints*, in which hICN is deployed only at the server and clients; (iii) called *hICN endpoints + edge*, in which hICN is deployed both at the endpoints and in the edge routers.

Table 3 compares the load at ATS in scenarios (i)-(ii) under the three client populations. Results show that using TCP/IP, the total number of requests handled by ATS grows linearly with the amount of clients, while using hICN it grows linearly with the number of channels being watched. The reason for such different behaviour lies in the content awareness brought by ICN at network layer and in the content-based rather than host-based communication model: requests for the same channel are directly satisfied by the hICN forwarder, reducing the load on the application. This is also confirmed by the absence of hits in the ATS cache using hICN, since duplicated requests do not reach the application. Considering that in a real deployment the difference between the number of actively watched channels and the number of total watchers can differ by many orders of magnitude, hICN scalability benefit may be significant. Moreover, hICN considerably reduces memory and CPU consumption in the system. Quantitative results are reported in Table 3.

metric	N=100 [C=22]		N=200 [C=30]		N=300 [C=35]	
	IP/TCP	hICN	IP/TCP	hICN	IP/TCP	hICN
r	743.4	215.7	1447.5	312.8	1963.0	358.7
hit	294.8	0	614.4	0	768.6	0
$miss$	448.5	215.7	832.2	312.8	1192.1	358.7
mem	1313.7	47.8	1484.6	47.1	1522.6	58.2
$\overline{\text{cpu}}$	9.6	6.1	16.7	6.1	21.4	6.4

r: #requests ($.10^3$), hit: cache hits ($.10^3$), $miss$: cache misses ($.10^3$), mem: total memory (MB) and $\overline{\text{cpu}}$: avg cpu usage (%)

Table 3: ATS performance metrics with N clients. C is the average number of channels being watched.

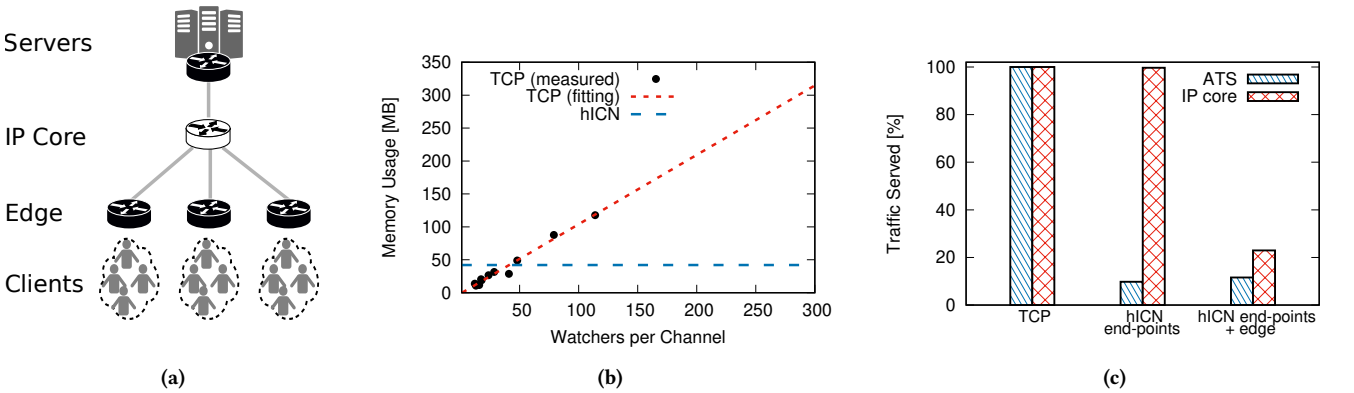

Figure 3: (a) Video distribution network used in Sec. 5.5. (b) hICN and TCP memory used by a single channel increasing the number of watchers. (c) Percentage of the traffic served by each component in the network.

Figure 3b compares the memory consumed by TCP and by hICN sockets for handling a single video channel with different number of watchers. The considered scenarios are (i)-(ii), as for Table 3. Each dot reports the memory consumed by the kernel to handle TCP sockets opened by each client. We measure the memory used by the sockets exploiting Linux proc filesystem (/proc/net/sockstats). The red dashed line shows the expected memory consumption of TCP when increasing the number of watchers. Such line is obtained fitting the TCP memory consumption we measured in our tests. The blue dashed line reports the memory cost of hICN socket to handle one video channel. This value corresponds to the amount of memory reserved in the hICN forwarder packet cache for each channel. As expected, results show that the memory required by TCP increases with the number of clients, while the memory required by the hICN socket remains constant. In fact, hICN socket does not maintain per-consumer connections, while TCP requires one socket for each connected client, so increasing the memory requirement. Figure 3c shows the additional improvement in terms of IP core traffic offload provided by hICN enablement at edge routers. The test considers 300 clients. The figure reports the percentage of total traffic received by clients that is served by ATS (in red) or by IP core network (in blue), respectively. The plot confirms what observed in Table 3: the request aggregation feature of hICN allows to reduce the amount of traffic served by ATS in scenario (ii) (*hICN end-points* case) w.r.t. the traffic served by ATS in scenario (i) (TCP/IP case). Deploying hICN also at the edge routers (scenario (iii) - *hICN end-points + edge*) reduces network traffic in the IP core network. It is worth noticing that IP core traffic is higher than the traffic received at ATS because of the considered network topology. The three hICN routers aggregate client requests and receive traffic independently from the server, possibly introducing multiple copies of the same interest/data packets.

6 RELATED WORK

Among ICN deployment strategies we can classify them in two broad classes: *full integration* and *partial integration* proposals.

Full integration. In the following, we discuss pros and cons of the main classes of full ICN deployment options [71]:

▪ **ICN as an overlay** – Envisaging the same transition model as IPv4-to-IPv6, the common deployment proposed for ICN is as an overlay (also "tunneling approach" in [71]): the new ICN protocol stack is transported on top of IP between pre-identified adjacent ICN routers, hereby creating islands of ICN deployments connected to each other via ICN/IP tunnels over existing IP-based infrastructure. To improve connectivity and control within and across ICN domains in terms of reliability and of scalability, different SDN-based approaches, and more specifically OpenFlow extensions, have been proposed for ICN deployment as an overlay on top of IP [34, 90, 91, 104]. While it prospects a rapid and easy deployment of ICN in fixed and in mobile networks [87], such deployment configuration requires the standardization of ICN packet format and protocols and, depending on the scale of the ICN deployment, the interoperability between IP and ICN routing protocols.

▪ **ICN as an underlay** – To overcome the limitations of overlay approaches via a native but scoped integration of ICN, proposals have emerged for an ICN deployment as an underlay in given islands of existing IP-based networks (e.g., inside a CDN or edge IoT network) [88]. The connection to the rest of the Internet is guaranteed by gateways or proxies translating semantics from ICN to IP routing domains. Unfortunately, that also implies dual stack challenges and a long timescale for expected adoption of the new stack in network equipments.

▪ **ICN in a slice** – Recently advocated in 5G context, this approach leverages the advances of network virtualization to realize slicing of network (compute, storage, bandwidth) and spectrum resources among applications and introduces ICN for the support of specific services (e.g. low latency, mobile, caching-aided). In [73, 75] the authors suggest creation of service slices using both IP and ICN and discuss the requirements for ICN introduction using programmable data planes.

Partial integration. Other proposals share the spirit of hICN and have suggested to reuse existing protocols to integrate ICN features in IP/TCP/HTTP. However, they only consider a subset of ICN aspects, trading-off some of its benefits, and consequently inherit inefficiencies of the layers underneath.

▪ **ICN semantics in IP** – The following prior art considers ways to embed resource names into IP packets for name-based forwarding. [86] suggests embedding content names in the IPv6 destination address via a proxy mapping HTTP URLs to IPv6 addresses: the FQDN is resolved (through DNS) and mapped to the first 64 bits of the IP address, while the path section is hashed to form the second half of the IP address. Their idea is to inherit some IPv6 functionalities such as mobility and security, while preserving routing scalability thanks to the two-level hierarchy of names.
CLIP [44] proposes to reserve an IPv6 subnet for content, and to split a content name into publisher label and content label. The publisher label is mapped into source and destination addresses for standard IP forwarding, while the content label is inserted into an ICN header extension and recognized only by ICN-compliant routers. CLIP also suggests a data-centric security model based on IPSec (AH for signing and ESP for encryption), decoupling privacy, authenticity and integrity.
CONET project [34], which considers an SDN-based overlay deployment of ICN with OpenFlow extension in the long term, suggests as short term alternative to use new IP options to carry content-level information. This would require standardization and might suffer from packet drop by non-compliant transit routers.

Unlike [44, 86] that only inherit ICN naming and caching properties, neglecting ICN stateful forwarding and pull-based connectionless transport, [34] aims at preserving ICN transport model and thus requires data packets to flow back to the consumer in the reverse direction. Temporary PIT state is encoded in the packet, in a specific CONET header extension or within payload. This solution has major drawbacks, since it prevents routers from aggregating requests, estimating of the congestion status of a path or performing in-network loss and congestion control.

▪ **ICN semantics in TCP/HTTP** – All previously presented approaches require the introduction and standardization of IP header extensions, which might cause packets to be dropped by routers, or the introduction of new layer 4 or 7 protocols, to be deployed as an overlay on top of IP. A different class of proposals has suggested integration of ICN semantics into transport or application layer protocols.
[85] proposes to use a transparent opportunistic interception of traffic at layer 4 or 7, in order to implement content-level functionalities in TCP. Unfortunately, those operations are costly and deemed not to scale beyond the network edge.

[36, 69, 92] start from the observation that at application-level, HTTP shares some key aspects with ICN: data addressing by name, pull-based communication model and coupling of request routing with caching. Beyond the efforts to optimize translation of HTTP to ICN semantics in [92], the content-centric nature of HTTP has generated a long debate in the research community about the benefits of a network/transport-layer approach such as ICN versus application-layer HTTP-based approaches for content delivery. In [69], authors develop the thesis that HTTP is already a content-centric protocol, providing middlebox support in the form of reverse and forward proxies, and leveraging DNS to decouple names from addresses. In the same CDN context, [36] demonstrates that little additional benefits come from pervasive caching and nearest-replica routing features of ICN, while still paying the cost for its integration in existing IP infrastructure. All these proposals have the merit of raising the question of the incremental deployability of ICN. However, they mostly target in-network caching and request-to-cache routing features of ICN, trading off other - in our view - key aspects of ICN network/transport layer such as in-network multicast/broadcast, network-assisted loss recovery and congestion control and native mobility support. Our attempt with hICN is to make the full ICN approach incrementally deployable in existing IP networks, without compromising any of the ICN architectural principles and related potential benefits.

7 DISCUSSION AND CONCLUSION

Motivated by the importance of short to mid-term deployability of ICN, in this paper we have analyzed Hybrid-ICN, which has the ambition to bring ICN inside the Internet protocol. Unlike other proposals, hICN focuses on preserving ICN communication model at network and transport layer, to inherit all intrinsic good properties of ICN that past research has highlighted, such as native security, mobility support, dynamic hop-by-hop forwarding and agile multi-path/multi-source transport, coupled with in-network caching.

hICN aims at deploying ICN at the end-points and in a few points at the network edge, where beneficial, guaranteeing transparent interconnection with existing IP elements and reuse of IP routing and management protocols. In this paper we have extensively used the open source implementation of hICN in the Fast Data project and shown the feasibility and scalability of hICN core elements provided by the software prototype. We have then applied it to a linear video streaming use case to highlight the higher user experience, resulting from in-network loss control, seamless mobility and multi-source/multi-path support over hetnet access, with better usage of network and system resources.

In this paper we have presented an extensive assessment of the architecture and open source implementation of hICN which is a promising solution in the mid-term for operational networks in a variety of segments: residential, enterprise and data center. Moreover, other high-benefit use cases such as Real Time Communication, IoT, AR/VR or low-latency edge-computing can benefit from hICN transport enhancements. hICN design would still require additional integration to security and transport features in the end-points such as TLS, DTLS and also the novel MLS [24] protocols to support as many applications as possible. Moreover if hICN can exploit IP control plane for intermediate nodes, it still requires a novel management-plane service to securely provision name prefixes at the producer, for instance, by extending DHCP options [43].

8 ACKNOWLEDGMENTS

We thank David Ward for the technical discussions to design the Hybrid-ICN architecture. We also thank David Oran and Paul Polakos for reviewing an early version of the paper. Thanks to Cullen Jennings and team for the feedback while integrating hICN in Cisco collaboration technologies. Many thanks to WebEx engineering team for valuable feedback. We thank our shepherd Ken Calvert and the anonymous reviewers for their useful feedback.

REFERENCES

[1] IPv6 CIDR REPORT. http://www.cidr-report.org/v6/as2.0/.
[2] NSF/Intel, Partnership on Information-Centric Networking in Wireless Edge Networks (ICN-WEN). https://www.nsf.gov/pubs/2016/nsf16586/nsf16586.htm.
[3] Open Broadcaster Software (OBS). https://obsproject.com/.
[4] PeeringDB, The Interconnection Database. https://www.peeringdb.com/.
[5] The NDN project, Named-Data Networking Principles. https://named-data.net/project/ndn-design-principles/.
[6] The NGINX HTTP server. https://nginx.org/en/.
[7] The NGINX RTMP module. https://nginx.org/en/.
[8] The pache Traffic Server, The Apache Foundation. http://trafficserver.apache.org/.
[9] The Route Views Project. http://www.routeviews.org/.
[10] YouTube Best Practice, Live encoder settings, bitrates, and resolutions. http://goo.gl/tDtc1i.
[11] Internet AS-level Topology Archive. http://irl.cs.ucla.edu/topology, 2018.
[12] MGEN, The multi-protocol generator. https://www.nrl.navy.mil/itd/ncs/products/mgen, 2018.
[13] Team Cymru, IP to ASN mapping. http://www.team-cymru.org/Services/ip-to-asn.html, Accessed Dec. 2017.
[14] AbdAllah, E. G., Hassanein, H. S., and Zulkernine, M. A survey of security attacks in information-centric networking. *IEEE Communications Surveys Tutorials 17*, 3 (thirdquarter 2015), 1441–1454.
[15] Adhatarao, S. S., Chen, J., Arumaithurai, M., Fu, X., and Ramakrishnan, K. K. Comparison of naming schema in icn. In *Proc. IEEE LANMAN'16* (June 2016).
[16] Atkinson, R., and Bhatti, S. Identifier-Locator Network Protocol (ILNP) Architectural Description. RFC 6740, Nov 2012.
[17] Augé, J., and al. libparistraceroute. https://github.com/libparistraceroute/.
[18] Augé, J., Carofiglio, G., Enguehard, M., Muscariello, L., and Sardara, M. Simple and efficient icn network virtualization with vicn. In *Proceedings of the 4th ACM Conference on Information-Centric Networking* (New York, NY, USA, 2017), ICN '17, ACM, pp. 216–217.
[19] Augé, J., Carofiglio, G., Grassi, G., Muscariello, L., Pau, G., and Zeng, X. Map-me: Managing anchor-less producer mobility in information-centric networks. *IEEE Transactions on Network and Service Management* (2018).
[20] Auge, J., Carofiglio, G., Muscariello, L., and Papalini, M. Anchorless mobility management through hICN (hICN-AMM): Deployment options. Internet-Draft draft-auge-dmm-hicn-mobility-deployment-options-01, Internet Engineering Task Force, Dec 2018. Work in Progress.
[21] Auge, J., Carofiglio, G., Muscariello, L., and Papalini, M. Anchorless mobility through hICN. Internet-Draft draft-auge-dmm-hicn-mobility-01, Internet Engineering Task Force, Dec 2018. Work in Progress.
[22] Barford, P., Bestavros, A., Byers, J., and Crovella, M. On the marginal utility of network topology measurements. In *Proc. ACM IMW'01* (2001).
[23] Bari, M. F., Chowdhury, S. R., Ahmed, R., Boutaba, R., and Mathieu, B. A survey of naming and routing in information-centric networks. *IEEE Communications Magazine 50*, 12 (December 2012), 44–53.
[24] Barnes, R., Millican, J., Omara, E., Cohn-Gordon, K., and Robert, R. The Messaging Layer Security (MLS) Protocol. Internet-Draft draft-ietf-mls-protocol-04, Internet Engineering Task Force, Mar 2019. Work in Progress.
[25] Baugher, M., Davie, B., Narayanan, A., and Oran, D. Self-verifying names for read-only named data. In *Proc. of IEEE INFOCOM 2012 Workshops* (March 2012), pp. 274–279.
[26] Bhat, D., Wang, C., Rizk, A., and Zink, M. A load balancing approach for adaptive bitrate streaming in information centric networks. In *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)* (June 2015), pp. 1–6.
[27] Carofiglio, G., Gallo, M., and Muscariello, L. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In *ACM ICN'12* (New York, NY, USA, 2012), pp. 37–42.
[28] Carofiglio, G., Gallo, M., and Muscariello, L. On the performance of bandwidth and storage sharing in information-centric networks. *Computer Networks 57*, 17 (Dec 2013), 3743–3758.
[29] Carofiglio, G., Gallo, M., and Muscariello, L. Optimal multipath congestion control and request forwarding in information-centric networks: Protocol design and experimentation. *Computer Networks 110* (2016), 104–117.
[30] Carofiglio, G., Gallo, M., Muscariello, L., and Perino, D. Pending interest table sizing in named data networking. In *Proceedings of the 2Nd ACM Conference on Information-Centric Networking* (2015), ACM-ICN '15, pp. 49–58.
[31] Carofiglio, G., Morabito, G., Muscariello, L., Solis, I., and Varvello, M. From content delivery today to information centric networking. *Comput. Netw. 57*, 16 (Nov 2013), 3116–3127.
[32] Carofiglio, G., Muscariello, L., Papalini, M., Rozhnova, N., and Zeng, X. Leveraging icn in-network control for loss detection and recovery in wireless mobile networks. In *ACM ICN'16* (New York, NY, USA, 2016), pp. 50–59.
[33] Chai, W., He, D., Psaras, I., and Pavlou, G. Cache "less for more" in information-centric networks. In *IFIP'12* (Berlin, Heidelberg, 2012), pp. 27–40.
[34] Detti, A., Salsano, S., and Blefari-Melazzi, N. Ip protocol suite extensions to support conet information centric networking. Internet-Draft draft-detti-conet-ip-option-05, Internet Engineering Task Force, Jun 2013. Work in Progress.
[35] Farinacci, D., Fuller, V., Meyer, D., and Lewis, D. The Locator/ID Separation Protocol (LISP). RFC 6830, Jan 2013.
[36] Fayazbakhsh, S., Lin, Y., Tootoonchian, A., Ghodsi, A., Koponen, T., Maggs, B., Ng, K., Sekar, V., and Shenker, S. Less pain, most of the gain: Incrementally deployable icn. In *Proc. of the ACM SIGCOMM 2013* (New York, NY, USA, 2013), SIGCOMM '13, pp. 147–158.
[37] Feng, B., Zhou, H., and Xu, Q. Mobility support in named data networking: a survey.
[38] Garcia-Luna-Aceves, J. J., Martinez-Castillo, J. E., and Menchaca-Mendez, R. Routing to multi-instantiated destinations: Principles, practice and applications. *IEEE Transactions on Mobile Computing PP*, 99 (2017), 1–1.
[39] Ghali, C., Tsudik, G., and Wood, C. (the futility of) data privacy in content-centric networking. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society* (2016), pp. 143–152.
[40] Ghali, C., Tsudik, G., and Wood, C. A. Network names in content-centric networking. In *Proc. of the 3rd ACM SIGCOMM ICN* (New York, NY, USA, 2016), ACM ICN'16, pp. 132–141.
[41] Ghodsi, A., Koponen, T., Rajahalme, J., Sarolahti, P., and Shenker, S. Naming in content-oriented architectures. In *ACM ICN'11* (New York, NY, USA, 2011), pp. 1–6.
[42] Grandl, R., Su, K., and Westphal, C. On the interaction of adaptive video streaming with content-centric networking. In *Proc. of IEEE Int. Packet Video Workshop* (Dec. 2013).
[43] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and Krishnan, S. Guidelines for Creating New DHCPv6 Options. RFC 7227, May 2014.
[44] Heath, L., Owen, H., Beyah, R., and State, R. Clip: Content labeling in ipv6, a layer 3 protocol for information centric networking. In *Proc. of ICC 2013* (June 2013), pp. 3732–3737.
[45] Hesmans, B., Duchene, F., Paasch, C., Detal, G., and Bonaventure, O. Are tcp extensions middlebox-proof? In *Proc. of the 2013 HotMiddlebox Workshop* (New York, NY, USA, 2013), HotMiddlebox '13, pp. 37–42.
[46] Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., and Tokuda, H. Is it still possible to extend tcp? In *Proc. of ACM SIGCOMM IMC 2011* (New York, NY, USA, 2011), IMC '11, pp. 181–194.
[47] Ioannidis, S., and Yeh, E. Jointly optimal routing and caching for arbitrary network topologies. In *Proc. of the 4th ACM SIGCOMM ICN 2017* (2017), pp. 77–87.
[48] ITU-T. Study Group 13, Data aware networking (information centric networking) – Requirements and capabilities . https://www.itu.int/rec/T-REC-Y.3071-201703-P, Dec 2016.
[49] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. Networking named content. In *Proc. of the 5th ACM CoNEXT* (New York, NY, USA, 2009), CoNEXT '09, pp. 1–12.
[50] James, C., Halepovic, E., Wang, M., Jana, R., and Shankaranarayanan, N. K. Is multipath tcp (mptcp) beneficial for video streaming over dash? In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)* (Sept 2016), pp. 331–336.
[51] Jiang, X., Bi, J., and Wang, Y. What benefits does ndn have in supporting mobility. In *Proc. of IEEE Symposium on Computers and Communications (ISCC)* (June 2014), pp. 1–6.
[52] Koch, P. A DNS RR Type for Lists of Address Prefixes (APL RR). RFC 3123, Jun 2001.
[53] Langley, A., et al. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (New York, NY, USA, 2017), SIGCOMM '17, ACM, pp. 183–196.
[54] Larumbe, F., and Mathur, A. Facebook engineering - under the hood: Broadcasting live video to millions. http://goo.gl/gsFyqo.
[55] Lederer, S., et al. Adaptive streaming over content centric networks in mobile networks using multiple links. In *Proc. of IEEE ICC* (2013).
[56] Lederer, S., Mueller, C., Rainer, B., Timmerer, C., and Hellwagner, H. An experimental analysis of dynamic adaptive streaming over http in content centric networks. In *2013 IEEE International Conference on Multimedia and Expo (ICME)* (July 2013), pp. 1–6.
[57] Linux Foundation FD.io. CICN project, wiki page. https://wiki.fd.io/view/Cicn.
[58] Linux Foundation FD.io. White Paper - Vector Packet Processing - One Terabit Software Router on Intel Xeon Scalable Processor Family Server. https://fd.io.
[59] Liu, X., Trappe, W., and Zhang, Y. Secure name resolution for identifier-to-locator mappings in the global internet. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)* (July 2013), pp. 1–7.
[60] Majeed, M., Ahmed, S., Muhammad, S., Song, H., and Rawat, D. Multimedia streaming in information-centric networking: A survey and future perspectives. *Computer Networks 125* (2017), 103 – 121.
[61] Misra, S., Tourani, R., and Majd, N. Secure content delivery in information-centric networks: Design, implementation, and analyses. In *ACM ICN'13* (New

York, NY, USA, 2013), pp. 73–78.
[62] Mosko, M., Solis, I., and Wood, C. A. Content-Centric Networking (CCNx) Messages in TLV Format. RFC 8609, Jul 2019.
[63] Mosko, M., Solis, I., and Wood, C. A. Content-Centric Networking (CCNx) Semantics. RFC 8569, Jul 2019.
[64] Muscariello, L., Carofiglio, G., Auge, J., and Papalini, M. Hybrid Information-Centric Networking. Internet-Draft draft-muscariello-intarea-hicn, Internet Engineering Task Force, Jun 2019. Work in Progress.
[65] Ngai, E., Ohlman, B., Tsudik, G., Uzun, E., Wahlisch, M., and Wood, C. A. Can we make a cake and eat it too? a discussion of icn security and privacy. *SIGCOMM Comput. Commun. Rev. 47*, 1 (Jan 2017), 49–54.
[66] Paper, G. A. W. Understanding information-centric networking and mobile edge computing. http://www.5gamericas.org/files/1214/8175/3330/Understanding_Information_Centric_Networking_and_Mobile_Edge_Computing.pdf, Dec 2016.
[67] Perino, D., Varvello, M., Linguaglossa, L., Laufer, R., and Boislaigue, R. Caesar: A content router for high-speed forwarding on content names. In *2014 ACM/IEEE ANCS Symposium* (Oct 2014), pp. 137–147.
[68] Petrangeli, S., Bouten, N., Claeys, M., and Turck, F. D. Towards svc-based adaptive streaming in information centric networks. In *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)* (June 2015), pp. 1–6.
[69] Popa, L., Ghodsi, A., and Stoica, I. Http as the narrow waist of the future internet. In *Proc. of the 9th ACM SIGCOMM Hotnets Workshop* (New York, NY, USA, 2010), Hotnets-IX, pp. 6:1–6:6.
[70] Posch, D., Kreuzberger, C., Rainer, B., and Hellwagner, H. Using in-network adaptation to tackle inefficiencies caused by dash in information-centric networks. In *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming* (New York, NY, USA, 2014), VideoNext '14, ACM, pp. 25–30.
[71] Rahman, A., Trossen, D., Kutscher, D., and Ravindran, R. Deployment Considerations for Information-Centric Networking (ICN). Internet-Draft draft-rahman-icnrg-deployment-guidelines-05, Internet Engineering Task Force, Jan 2018. Work in Progress.
[72] Rainer, B., Posch, D., and Hellwagner, H. Investigating the performance of pull-based dynamic adaptive streaming in ndn. *IEEE Journal on Selected Areas in Communications 34*, 8 (Aug 2016), 2130–2140.
[73] Ravindran, R., Chakraborti, A., Amin, S., Azgin, A., and Wang, G. 5g-icn: Delivering icn services over 5g using network slicing. *IEEE Communications Magazine 55*, 5 (May 2017), 101–107.
[74] Ravindran, R., Lo, S., Zhang, X., and Wang, G. Supporting seamless mobility in named data networking. In *Proc. of IEEE ICC 2012* (June 2012), pp. 5854–5869.
[75] Ravindran, R., Suthar, P., and Wang, G. Enabling ICN in 3GPP's 5G NextGen Core Architecture. Internet-Draft draft-ravi-icnrg-5gc-icn-00, Internet Engineering Task Force, Oct 2017.
[76] Ren, Y., Li, J., Shi, S., Li, L., Wang, G., and Zhang, B. Congestion control in named data networking - a survey. *Comput. Commun. 86*, C (Jul 2016), 1–11.
[77] Rossini, G., and Rossi, D. A dive into the caching performance of content centric networking. In *IEEE 17th Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* (Sept 2012), pp. 105–109.
[78] Rossini, G., and Rossi, D. Evaluating ccn multi-path interest forwarding strategies. *Comput. Commun. 36*, 7 (Apr 2013), 771–778.
[79] Saino, L., Cocora, C., and Pavlou, G. Cctcp: A scalable receiver-driven congestion control protocol for content centric networking. In *Proc. of IEEE ICC 2013* (June 2013), pp. 3775–3780.
[80] Samain, J., Carofiglio, G., Muscariello, L., Papalini, M., Sardara, M., Tortelli, M., and Rossi, D. Dynamic adaptive video streaming: Towards a systematic comparison of icn and tcp/ip. *IEEE Transactions on Multimedia 19*, 10 (Oct 2017), 2166–2181.
[81] Sardara, M., Muscariello, L., and Compagno, A. A transport layer and socket api for (h)icn: Design, implementation and performance analysis. In *Proc. of ACM SIGCOMM ICN '18* (2018).
[82] Shalunov, S., Hazel, G., Iyengar, J., and Kühlewind, M. Low Extra Delay Background Transport (LEDBAT). RFC 6817, Dec 2012.
[83] Smetters, D., and Jacobson, V. Securing network content. Tech. rep., 2009.
[84] So, W., Narayanan, A., and Oran, D. Named data networking on a router: Fast and dos-resistant forwarding with hash tables. In *2013 ACM/IEEE ANCS Symposium* (Oct 2013), pp. 215–225.
[85] Srinivasan, S., Rimac, I., Hilt, V., Steiner, M., and Schulzrinne, H. Unveiling the content-centric features of tcp. In *Proc. of IEEE ICC 2011* (June 2011), pp. 1–5.
[86] Srinivasan, S., and Schulzrinne, H. Ipv6 addresses as content names in information-centric networking. In *USENIX ATC 2011 - Poster session* (June 2011).
[87] Suthar, P., Stolic, M., Jangam, A., and Trossen, D. Native Deployment of ICN in LTE, 4G Mobile Networks. Internet-Draft draft-suthar-icnrg-icn-lte-4g-04, Internet Engineering Task Force, Nov 2017. Work in Progress.
[88] Trossen, D., Reed, M. J., Riihijärvi, J., Georgiades, M., Fotiou, N., and Xylomenos, G. Ip over icn - the better ip? In *2015 European Conference on Networks and Communications (EuCNC)* (Jun 2015), pp. 413–417.
[89] Tyson, G., Sastry, N., Rimac, I., Cuevas, R., and Mauthe, A. A survey of mobility in information-centric networks: Challenges and research directions. In *Proc. of the 1st ACM NoM Workshop 2012* (New York, NY, USA, 2012), NoM '12, pp. 1–6.
[90] Vahlenkamp, M., Schneider, F., Kutscher, D., and Seedorf, J. Enabling ICN in IP networks using SDN. In *ICNP* (2013), pp. 1–2.
[91] van Adrichem, N. L. M., and Kuipers, F. Ndnflow: Software-defined named data networking. In *NetSoft* (2015), pp. 1–5.
[92] Wang, S., Bi, J., Wu, J., Yang, X., and Fan, L. On adapting http protocol to content centric networking. In *Proc. of the 7th International Conference on Future Internet Technologies* (New York, NY, USA, 2012), CFI '12, pp. 1–6.
[93] Wang, Y., Rozhnova, N., Narayanan, A., Oran, D., and Rhee, I. An improved hop-by-hop interest shaper for congestion control in named data networking. In *ACM ICN'13* (New York, NY, USA, 2013), pp. 55–60.
[94] Westphal, C., et al. Adaptive Video Streaming over Information-Centric Networking (ICN). RFC 7933, Aug 2016.
[95] White, G., and Rutz, G. Content delivery in content-centric networks. http://www.cablelabs.com/wp-content/uploads/2016/02/Content-Delivery-with-Content-Centric-Networking-Feb-2016.pdf, 2016.
[96] Yi, C., Afanasyev, A., Moiseenko, I., Wang, L., Zhang, B., and Zhang, L. A case for stateful forwarding plane. *Comput. Commun. 36*, 7 (Apr 2013), 779–791.
[97] Yi, C., Afanasyev, A., Wang, L., Zhang, B., and Zhang, L. Adaptive forwarding in named data networking. *SIGCOMM Comput. Commun. Rev. 42*, 3 (Jun 2012), 62–67.
[98] Yu, Y., Afanasyev, A., Clark, D., claffy, k., Jacobson, V., and Zhang, L. Schematizing trust in named data networking. In *Proc. of the 2Nd ACM SIGCOMM ICN* (New York, NY, USA, 2015), ACM ICN'15, pp. 177–186.
[99] Yuan, H., Crowley, P., and Song, T. Enhancing scalable name-based forwarding. In *ANCS* (2017), pp. 60–69.
[100] Zhang, F., Zhang, Y., Reznik, A., Liu, H., Qian, C., and Xu, C. A transport protocol for content-centric networking with explicit congestion control. In *Proc. of 23rd ICCCN 2014* (Aug 2014), pp. 1–8.
[101] Zhang, G., Li, Y., and Lin, T. Caching in information centric networking: A survey. *Computer Networks 57*, 16 (2013), 3128 – 3141.
[102] Zhang, M., Luo, H., and Zhang, H. A survey of caching mechanisms in information-centric networking. *IEEE Communications Surveys Tutorials 17*, 3 (2015), 1473–1499.
[103] Zhang, Y., Afanasyev, A., Burke, J., and Zhang, L. A survey of mobility support in named data networking. In *IEEE INFOCOM WKSHPS 2016* (April 2016), pp. 83–88.
[104] Zuraniewski, P., van Adrichem, N., Ravesteijn, D., IJntema, W., Papadopoulos, C., and Fan, C. Facilitating icn deployment with an extended openflow protocol. In *Proc. of the 4th ACM SIGCOMM ICN* (New York, NY, USA, 2017), ICN '17, pp. 123–133.