Scalable Content-centric Routing for Hybrid ICN

Sergi Rene, George Pavlou
University College London, UK
{s.rene, g.pavlou}@ucl.ac.uk

Onur Ascigil
Lancaster University, United Kingdom
o.ascigil@lancaster.ac.uk

***Abstract*—Hybrid Information-Centric Networking (hICN) is an incrementally-deployable information-centric networking architecture that is built on top of IPv6. In hICN, application-level identifiers are directly used to route interest packets (*i.e.*, request for content) to fetch a copy of the desired content/data from any location. However, following the Internet Protocol conventions that require storing pre-computed routing/forwarding state for all prefixes in the routers raises scalability concerns, especially at the inter-domain level. Here we consider instead the other extreme; i.e. on-demand routing computation for content name prefixes when interest packets arrive at the router. Following this approach, we propose a centralized routing service within a domain that keeps a mapping between hICN name prefixes and locators (*i.e.*, routable addresses) to hICN routers. Once a locator is received, an hICN router forwards an interest packet towards the intended destination using segment routing. We evaluated the proposed solution through a real testbed implementation in order to demonstrate that the performance is equivalent to typical hICN forwarding, while offering a scalability solution.**

***Index Terms*—hICN, Routing, Scalability**

I. Introduction

Information-Centric Networking (ICN) is an architectural approach to evolve the "point-to-point connectivity" service of the current Internet to "retrieve data with a given name" service [1], focusing on retrieving data by name, versus connecting to a specific point of the network, without caring from where the data is retrieved. In ICN, consumers retrieve a copy of the data/content by sending *interest* packets that carry the name of the desired data. The service semantics is such that data packets can be discovered and retrieved from any node in the network independent of its location. Most ICN architectures, such as Named Data Networking (NDN) [2], use a *hop-by-hop name-based routing mechanism* to forward interest packets towards data in the network. Using stateful forwarding mechanisms, name-based routing can support sophisticated strategies to discover content in the network [3], [4]. However, ICN large-scale testing and deployment in operational networks are yet to happen, mostly due to the lack of a clear incremental deployment strategy.

The main goal of the hybrid ICN (hICN) architecture [5] is to bring ICN capabilities into existing IP networks while supporting partial deployment, *i.e.*, working even when only a few strategic nodes are hICN enabled. More specifically, hICN integrates Information-Centric Networking (ICN) in IP and, unlike other proposals, it does not use encapsulation or tunnelling techniques, and also does not run as an overlay network. hICN by design intends to share the same infrastructure with regular IP traffic and reuses most of the parts of an IPv6 router, such as its FIB table, and with minor modifications, it provides information-centric capabilities, such as interest-based routing and forwarding and in-network caching. Moreover, hICN is 100% compatible with IPv6 traffic, and hICN packets can be forwarded by on-path IPv6 routers without any modifications. hICN encodes location-independent, application-level names in globally routable name prefixes in the form of /128 IPv6 addresses, called name prefixes. The name prefix is split into a routable prefix and a 64-bit data identifier. The name prefix is used to identify an application object, a service or in general an application-level source of data in the network. This is incarnated by a listening socket that binds to the name prefix. Along with the name prefix, a name suffix is used in hICN to index segmented data within the scope of the name prefix used by the application.

However, hICN inherits forwarding state scalability issues from ICN that must be addressed to be a viable networking solution. ICN routers need to maintain state information for all the content they have seen, which allows them to forward content efficiently. However, as the number of content items increases, the amount of forwarding state required by routers grows, and this impacts the scalability of the network. hICN achieves similar scalability properties with IP at the inter-domain level. However, routing (interest) packets within an Autonomous System (AS) requires awareness of which /128 hICN network addresses are leased to which hosts in order to route hICN interest packets to the correct hosts within the domain. This is a potential scalability issue because ASes can lease a large number of hICN name prefixes at distributed locations. With a limited Forwarding Information Base (FIB) in the forwarders, it is challenging to perform intra-domain routing of interests to a host that has a replica of the requested named content within an AS.

In this work, we propose an on-demand routing architecture, called hICN On-Demand Routing, that uses Segment Routing IPv6 (SRv6) as forwarding hints to route packets towards the locator learned from the central routing controller, named Routing Service (RS).

Segment Routing leverages IPv6 addresses as routing locators and allows for programmable forwarding through the insertion of SRv6 encapsulation in the packet header. SRv6 has the potential to provide a unified and scalable routing solution for hybrid ICN architectures by enabling content-centric routing in the ICN domain and IP-based routing in the IP domain through a single routing protocol.

Our solution leverages the content name as an identifier for routing in the ICN domain and the SRv6 segment as a locator

for routing in the IP domain. We present the design of our solution, including the mapping of content names to SRv6 segments and the implementation of our solution in a hybrid ICN testbed. We also evaluate the performance of our solution in terms of routing scalability, latency, and packet delivery ratio, comparing with hICN forwarding where no routing is required.

The rest of the paper is organized as follows. In Section II we discuss related work and then in Section III we present the on-demand routing mechanisms used in our solution. In Section IV, we detail the performance evaluation of our approach, and finally, we summarize and discuss future work in Section V.

II. Related Work

To deal with the ICN forwarding state scalability problem in the Named Data Networking (NDN) architecture, Afanasyev et al [6] proposes a namespace mapping mechanism whereby content name prefixes are translated to network location names using a DNS-like mapping service. End-users include location names (in addition to content names) as a forwarding hint for the routers in their interest packets to be used in case the content name prefix is not found in the routers' FIB. On the other hand, in this work we propose an in-network mapping service without reliance on external mapping services.

We extend our earlier scaling proposal [7] based on a recent research project with Cisco which allowed us to implement it in a real testbed and apply it to the hICN architecture. Our approach is similar in spirit to both [8] and [9], where the authors discuss a mapping service that provides forwarding hints, within a network domain, mapping global content names to routable identifiers or locators. But we apply the mapping concept to an actual hICN architecture using real system tools and mechanisms in order to evaluate if our scalable forwarding maintains the high performance of standard hICN forwarding.

III. On-demand routing for Hybrid ICN

Current routing technology can support routing entries in the order of millions of routes (see [10]), which is orders of magnitude smaller than the expected number of routes in information-centric routing, i.e., $\approx 10^9$ name prefixes [9], [11]. Therefore, storage of routing and forwarding information for a large content namespace at a single forwarding node is not realistic. Instead, in this paper, we propose an hICN On-Demand Routing (hODR) scheme following a centralised software-defined approach, where routing information of each domain[1] is collected and maintained by a local domain service, namely the Routing Service (RS). In hODR, routers retrieve AS-specific routing information in the form of SRv6 Segment Identifiers (SID) from the RS and perform on-demand routing as interests arrive. Routing information retrieved from RS can be cached in a local data structure called Routing Information Store (RIS) in each hODR router.

In traditional network architectures, the data plane, which is responsible for forwarding packets, and the control plane, which is responsible for preparing the forwarding state, coexist within network devices deployed as a single integrated system. In centralised software-defined routing, the different planes are split into logical entities allowing programmatic management, control, and optimization of network resources from third devices.

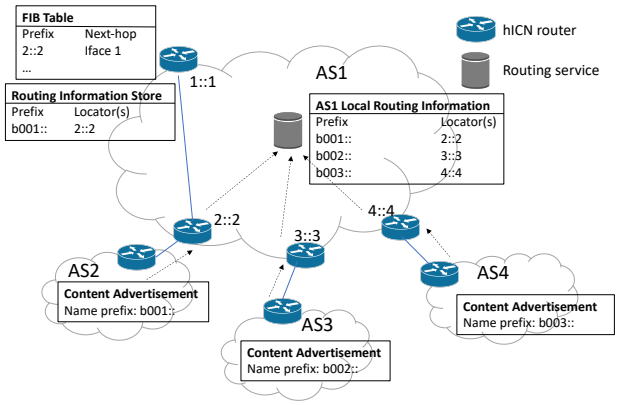


Fig. 1: On demand routing architecture

In an hICN environment, routing state is provided by IPv6 routing protocols, such as Open Shortest Path First (OSPFv6) protocol. Control plane separation allows the implementation of an on-demand routing protocol that reduces the amount of routing state that routers should store, providing a scalable routing solution in the face of the expected explosion of network routes with the deployment of information-centric solutions. Moreover, such an approach enables network administrators to dynamically adjust routing policies in response to changing network conditions, ensuring optimal routing performance.

In Figure 1 we can observe an overview of the proposed architecture. We assume that globally propagated content advertisements originate from content owners and also possibly from persistent storage systems (e.g., CDNs) that can act as producers. We assume the existence of a name-based, BGP-like inter-domain protocol and a location-based intra-domain protocol (e.g., OSPF [12]). Routable addresses (*i.e.*, locator) advertisements are disseminated (e.g. 1::1, 2::2) within the domain, but also local content name prefixes from local AS producers. Content prefix advertisements arriving or originating within a domain are sent directly to the RS by the border router of the domain. The RS acts as a storage of global routing information where it maps hICN names to intra-domain locator addresses. These locators are border router addresses, in case of external AS content, or the router closer to the producer in case of content produced in the same AS domain. Once a router receives an interest containing an hICN name prefix, in case there is no previously cached information in the RIS for the name prefix, the router queries the RS. The RS replies with the locator of the border router towards the destination AS for the name prefix. The routing information is cached in the RIS to be used for other arriving interests towards the same prefix. The router uses Segment Routing to route the hICN

[1]We use Autonomous System (AS) and domain interchangeably in the rest of the paper.

interest towards the border router connecting to the AS that is the source of the content, or the closest router in case local content is produced in the same AS. The following subsection details all the hICN interest forwarding processes at the router level.

A. hICN routing and forwarding

The forwarding path in hICN is composed of the interest and data path. Interests are generated by clients to request specific data, and data packets are generated by producers containing the content requested. In the following, we describe how interests and data packets are processed in vanilla hICN and we compare it with our proposed hICN On-Demand Routing.

Interests and data are processed at the hICN node in a different way. hICN routers can cache both interests and data packets. Interests require to be cached to transmit the corresponding data packet back to the consumer (*i.e.*, source) as the source address field in the interest contains the interface identifier of the hICN node having transmitted the interest. Data packets are optionally cached if needed. By caching data packets, hICN routers can replace data origin (producers) by providing the data from the cache, therefore saving bandwidth and improving network performance.

hICN interest forwarding path is based on lookups in the IP FIB just like any other IP packet, with the additional processing due to a cache lookup to check if the actual reply is already present in the local cache.

When an interest packet is received in a router, the forwarding steps are the following:

1) The incoming interest packet is parsed to obtain the name prefix and the name suffix. An exact match look-up is made in the packet cache using the full packet name as the key. If there is a match the data packet is transmitted with the destination address of the data packet equal to the source address of the interest packet.
2) In case no data packet is matched, there is a lookup for an interest previously cached. In case there is an interest with the same source address, the interest is classified as a duplicate and dropped. In case it has a different source address, the interest is cached but filtered.
3) In case there is no match with a cached interest, it is forwarded to the next hop. To determine the next hop, the router passes the interest to the egress and a traditional lookup in the IPv6 FIB table is done. The source address of the interest packet is replaced with the address of the egress router interface.
4) The procedure is repeated by each hop until the interest reaches the producer. Once the interest reaches the producer, a data packet is sent towards the source node of the interest. The destination address of the data packet is the source address of the interest received.
5) At each hICN router crossed, the data packet is optionally cached and there is a lookup for all the interests matched with the data packet. A data packet is sent for all the interests cached with the destination address set to each of the interest source addresses, till it reaches the interest sender.

In Figure 2 we present the interest packet processing workflow in an hICN router. In orange, we depict the actions made by a vanilla hICN router and in green the new actions added by a hODR router:

1) When there is no match for an hICN interest, there is no Segment Routing (SRv6) header in the interest packet, and there is no information in the RIS for the specific prefix, a lookup to the RS is done.
2) In case there is a match with the name prefix in the RS, the RS replies with a locator that identifies a border router of the AS.
3) In case there is no match in the RS, a message is sent to either drop the interest packet or sent it towards a default route, depending on the policy configured in the RS.
4) Once the hODR router receives the locator address, it stores the information in the RIS table and it creates a new policy that encapsulates any interest that matches the prefix into an SRv6 packet with the destination address of the locator address.
5) When hICN routers receive SRv6 packets with an hICN interest, the original interest is stored with the hICN prefix disregarding the SRv6 destination and replacing the hICN source address with the router egress interface address. In the case of traversing non-hICN IPv6 routers, interests are forwarded towards the destination address in the SRv6 header, without caring about any hICN instruction.
6) Intermediate hICN routers cache the hICN to SRv6 destination address mapping in the RIS, so when new interests arrive for the name prefix, no need to query the RS.
7) Border routers decapsulate SRv6 packets (using SRv6 END.DX6 network function [13]) towards next AS destination.
8) Data packets follow the interest path following the usual interest matching of all the interests cached along the path, following the usual hICN functionality with no encapsulation.

B. Prototype implementation

All the elements of the hODR architecture have been implemented in a prototype and all the software is available in this repository[2], including the extension to the hICN router, the routing service and the configuration files required for the testbed setup presented in IV.

Our solution is incrementally deployable and is compatible with any hICN router that supports the protocols detailed below, which are supported by default in the available hICN implementation [3].

For the hICN router we extended the available plugin[4] for the Vector Packet Processing (VPP) router. The VPP platform is an extensible framework that provides a software-based

[2]https://github.com/srene/hicn-scalable-roting

[3]https://hicn.readthedocs.io/en/latest/vpp-plugin.html

[4]

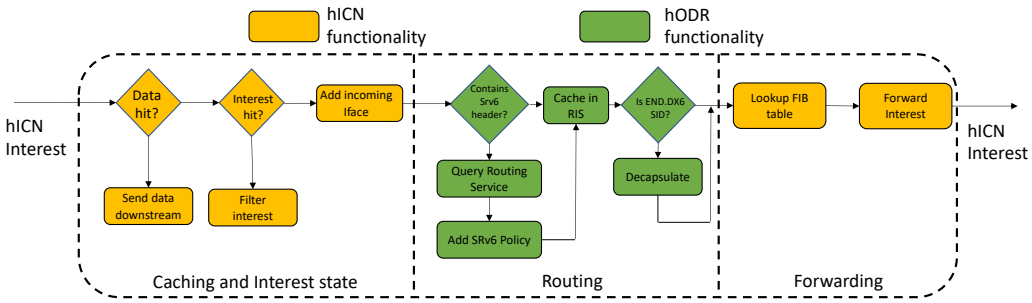


Fig. 2: Interest packet processing in an hODR router

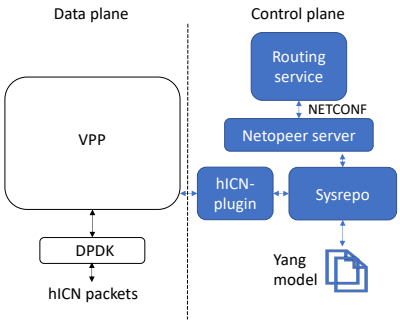


Fig. 3: Node architecture

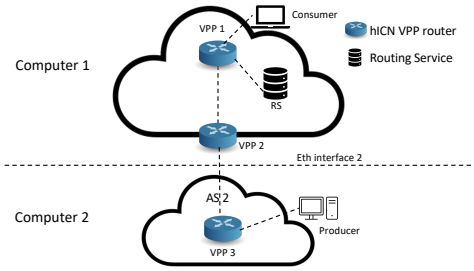


Fig. 4: Testbed

router that can run on any commodity server, based on the open source version of Cisco's Vector Packet Processing (VPP) technology, a high-performance, packet-processing stack. VPP is part of the Fast Data Project[5]. VPP uses the Data Plane Development Kit (DPDK[6]) device drivers and libraries for many of its layer-1 functions.

For this project, we extended the available VPP hICN plugin, that enables hICN forwarding for a VPP router, with the support for on-demand routing. The changes made in the code were focused on adapting the VPP hICN plugin to make it compatible with SRv6 traffic, detecting encapsulated packets and routing interests accordingly. We also adapted virtual interfaces creation – to route back data packets towards the source, following interests' state, and automatically enable hICN prefix routes for the data path to enable caching.

For the RS, we implemented a controller in Python that interacts with the VPP router using Network Configuration Protocol (NETCONF) [14]. NETCONF is a network management protocol that was created to overcome the limitations of SNMP. NETCONF operates a manager and a device. The managed device is represented using Yet Another Next Generation language (YANG) and configuration changes are applied by the NETCONF manager transmitting messages using XML. NETCONF protocol is used in the southbound interface, to connect the data plane and the control plane.

For the communication between the RS and the hODR router using NETCONF, we also extended the original VPP API and we implemented a new Sysrepo [7] plugin. Sysrepo is a YANG-based datastore for Unix/Linux systems. Therefore, we used Sysrepo to load the YANG-based model in the system and we interact with the VPP router using that model and NETCONF protocol. We built a sysrepo plugin that receives NETCONF actions defined in the YANG model and interacts with the VPP router using the extended API. Along with Sysrepo, we used a Netopeer2 server[8], that is used to enable NETCONF communications with another endpoint. In Figure 3 we show how the different elements interact between them.

IV. Performance Evaluation

In this section, we evaluate the performance of the proposed hODR scheme on a range of parameters. The objective is to evaluate the performance of hODR in terms of additional latency in information retrieval and throughput achieved under different configurations in a real testbed using the prototype we built using the software described in the previous section. Next, we describe the setup of our evaluations, before presenting the experiments in the remaining sections.

For the evaluation of the hODR, we used a testbed deployed in two different computers. The testbed uses two Linux workstations connected through a 10Gbps network Ethernet link. Each computer has an AMD Threadripper 3960x 3.8 GHz processor with 24 cores, 128MB (L3), 12MB (L2) cache, and 256GB DDR4 memory. Each workstation has a 10Gbps ASUS XG-C100C Ethernet interface.

[5]https://fd.io/
[6]https://www.dpdk.org/
[7]https://www.sysrepo.org/
[8]https://github.com/CESNET/netopeer2

Testbed set-up: We use the network topology depicted in Figure 4. In the topology, we basically represent two different AS domains. Each computer represents one domain. In the first computer, we deploy two different router instances, VPP1 and VPP2. VPP1 acts as a consumer agent and generates interest packets to request data with different name prefixes, using a VPP packet generator client. VPP1 is connected to VPP2 using virtual shared memory packet (memif) interfaces[9] and acting as a border router of the domain. VPP2 has an egress interface configured using the Ethernet interface directly connected to the second computer. This interface connects to AS2, being the 10Gbps interface with the maximum bandwidth of the network. In the second domain AS2, there is a single VPP instance deployed, VPP3. VPP3 is directly connected to VPP2 through its Ethernet interface. VPP3 acts as hICN producer and generates data packets to reply to requested data with the different name prefixes configured, using a VPP packet generator server. There is a Routing Service (RS) deployed in Computer 1. The RS is deployed using a Docker[10] container image and is connected to VPP1 using NETCONF and the sysrepo plugin detailed in III-B. The link latency between RS and VPP1 is set using Linux tool for traffic management (tc)[11], and is configured according to the parameters detailed in the next paragraph.

TABLE I: Default evaluation parameters.

Parameter	Value
Number of name prefixes ($\|P\|$)	1M
$\|RIS\|/\|P\|$	0.0075
Name prefix popularity (Zipf exponent)	1.0
RS latency	20ms
Router link latency	1ms

Parameters: We focus mainly on the parameters related to the scalability of routing and forwarding performance. These are the access distribution (i.e., popularity) of name prefixes, the size of the RIS on each forwarder and the latency between routers and the RS. Based on the assumption that the global prefix size is 10^9, we extrapolate a realistic ratio of the size of a forwarder's RIS cache to the global prefix size (i.e., $|RIS|/|P|$) using technical specifications of reasonably powerful routers. In particular, a BGP router can store $7.5M$ routes in its FIB table with the current technology [10]. This results in $|RIS|/|P|$ ratio of 0.0075, and we use this as the default ratio in the simulations. We do not limit RS storage space, since we assume that the RS can provide scalable storage for global routing information within the domain through horizontal scaling. Also, we set the default size of data packets cache (i.e., used only for caching content) of each forwarder to zero in order to focus on the performance of routing information discovery.

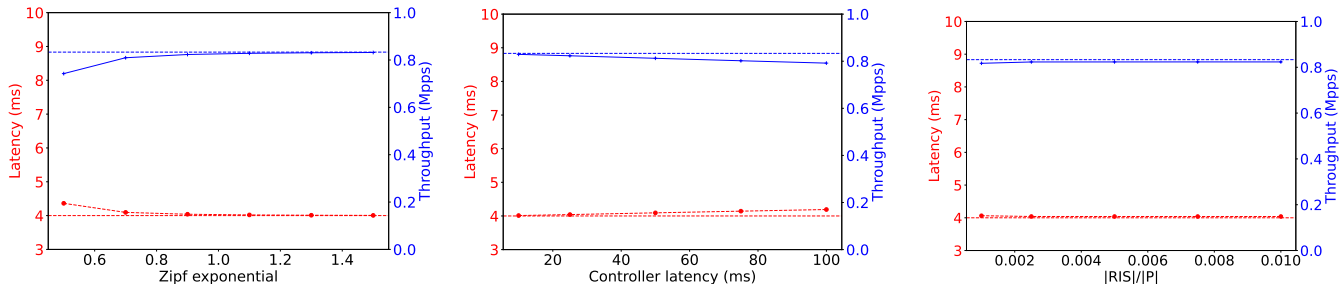
Table I provides the default simulation parameters for the experiments. The first parameter is the number of different content items (i.e., name prefixes) used in the evaluation. We limit the size of name prefix at $|P| = 1M$ for all the experiments. The content name popularity is modelled using a Zipf distribution of exponential 1.0 by default, following the distribution of content name popularity that has been measured in different contexts (e.g., web caching) [15], [16]. For simplicity, each content item fits in a single 1500 bytes packet in the evaluation. The links between routers are configured to 1 ms latency and the default latency between VPP1 and RS is set to 20 ms.

Our evaluation is based on the following metrics in order to prove the feasibility of an on-demand routing service for hICN, and how close we can be to the ideal case where all routes are previously known and there is no added latency:

- **Throughput** (in packets per second): We measure the packet rate received at VPP1 for all the content requested. We aim to check whether performance in the router can be close to the line rate, equivalent to 10Gbps, but measured in packets per second.
- **Packet latency** (in milliseconds): This metric measures the average round-trip time (RTT) delay in retrieving content per issued interest. The RTT delay includes the amount of time it takes for the forwarders to retrieve routing information when there is a RIS miss at the first hop forwarder from the end user. The object is to observe whether there is a significant deviation of the average measured RTT to the minimum latency (RTT measured when the RS is not involved in the forwarding), equivalent to 4 ms.

In Figure 5(a) we can observe the throughput measured and the total packet retrieval latency (RTT) when using different exponential in the Zipf distribution that models the content requested. We observe that for low exponentials (which means less content with high popularity), and therefore more requests to the RS, there is an impact on the performance compared with the optimal values, shown as a horizontal dashed line, equivalent to 0.833 Mpps for the throughput, and 4 ms for the RTT. However, for Zipf exponentials of 0.8 or higher, there is no impact observed in the network performance. In Figure 5(b) we can observe the throughput measured and the total packet retrieval latency (RTT) when using different latencies in the link between VPP1 and the RS. The latency is set between 10 ms and 100 ms. We can observe the performance is close to the optimal, shown as a dashed horizontal line, only observing a deviation for high latencies to the RS, close to 100 ms, which is equivalent to latency between different continents, unrealistic for latency within the same AS domain. In Figure 5(c) we can observe the throughput measured and the total packet retrieval latency (RTT) with different configurations of the size of the RIS. The RIS size is set to $|RIS|/|P|$ ratio between 0.001 and 0.01. Taking into account we evaluated the performance for a $1M$ content size, this is equivalent of a RIS size between 1000 and 10000 entries. We observe the RIS size is barely affecting the performance and in any of the configurations evaluated the throughput and the latency is very close to the optimal value, which is shown as a horizontal dashed line. This means that even with a small RIS size, a hODR router is able to perform without a problem regarding the amount of memory used to store RIS entries, for the

[9]https://doc.dpdk.org/guides/nics/memif.html
[10]https://www.docker.com/
[11]https://man7.org/linux/man-pages/man8/tc.8.html



(a) Load distribution (b) Routing service latency (c) Router RIS size

Fig. 5: hODR routing service measurements

evaluation parameters used.

V. Summary and Future Work

In this paper, we present an on-demand routing solution for hICN. Instead of pre-computing routes for content name prefixes, we propose a centralized routing service within a domain that keeps a mapping between hICN name prefixes and locators (routable addresses to hICN routers). Once a locator is received for the requested hICN content prefix, an hICN router forwards an interest packet towards the intended destination using SRv6 following an centralised software-based approach.

In this work, we demonstrate that with a modest amount of storage at each forwarder to cache routing information, we can perform purely on-demand routing with reasonable bandwidth and latency overheads. In order to limit the overheads, we exploit the locality of reference in the destination routable prefixes contacted by users through caching and discovery of routing information.

As a future work, we plan to investigate the performance of hODR routers in more complex scenarios, using large-scale ISP topologies We also plan to investigate hybrid mechanisms combining pre-computation and on-demand routing mechanisms. In that case, forwarders can use their slower memory (e.g., DRAM) to store less popular routing information and cache pre-computed forwarding information for popular content in their fast memory.

Acknowledgment

This work has been supported by the Cisco grant number 1923388.

References

[1] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[3] O. Ascigil, V. Sourlas, I. Psaras, and G. Pavlou, "A native content discovery mechanism for the information-centric networks," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 145–155.

[4] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiaseelan, and J. Crowcroft, "Pro-diluvian: Understanding scoped-flooding for content discovery in information-centric networking," in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*. ACM, 2015, pp. 9–18.

[5] G. Carofiglio, L. Muscariello, J. Augé, M. Papalini, M. Sardara, and A. Compagno, "Enabling icn in the internet protocol: Analysis and evaluation of the hybrid-icn architecture," in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, 2019, pp. 55–66.

[6] A. Afanasyev, C. Yi, L. Wang, B. Zhang, and L. Zhang, "Snamp: Secure namespace mapping to scale ndn forwarding," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*. IEEE, 2015, pp. 281–286.

[7] O. Ascigil, S. Rene, I. Psaras, and G. Pavlou, "On-demand routing for scalable name-based forwarding," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, ser. ICN '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 67–76. [Online]. Available: https://doi.org/10.1145/3267955.3267968

[8] Y. Zhang, Z. Xia, A. Afanasyev, and L. Zhang, "A note on routing scalability in named data networking," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2019, pp. 1–6.

[9] A. Detti, M. Pomposini, N. Blefari-Melazzi, and S. Salsano, "Supporting the web with an information centric network that routes by name," *Computer Networks*, vol. 56, no. 17, pp. 3705–3722, 2012.

[10] Cisco, "Cisco ASR 1000 Series Route Processor Data Sheet," https://www.cisco.com/c/en/us/products/collateral/routers/asr-1000-series-aggregation-services-routers/data_sheet_c78-441072.html, 2018, online; accessed 16 March 2023.

[11] T. Song, H. Yuan, P. Crowley, and B. Zhang, "Scalable name-based packet forwarding: From millions to billions," in *Proceedings of the 2nd ACM conference on information-centric networking*. ACM, 2015, pp. 19–28.

[12] J. Moy, "Ospf version 2," Internet RFC 2328, April 1998.

[13] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming," RFC 8986, Feb. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8986

[14] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, "Network Configuration Protocol (NETCONF)," RFC 6241, Jun. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6241

[15] M. Halvey, M. T. Keane, and B. Smyth, "Mobile web surfing is the same as web surfing," *Communications of the ACM*, vol. 49, no. 3, pp. 76–81, 2006.

[16] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in web client access patterns: Characteristics and caching implications," *World Wide Web*, vol. 2, no. 1-2, pp. 15–28, 1999.