Managing the Evolution to Future Internet Architectures and Seamless Interoperation

Mohammad Jahanian*, Jiachen Chen†, and K. K. Ramakrishnan*
*University of California, Riverside, CA, USA. Email: mjaha001@ucr.edu, kk@cs.ucr.edu
†WINLAB, Rutgers University, NJ, USA. Email: jiachen@winlab.rutgers.edu

***Abstract*—With the increasing diversity of application needs (datacenters, IoT, content retrieval, industrial automation, *etc.*), new network architectures are continually being proposed to address specific and particular requirements. From a network management perspective, it is both important and challenging to enable evolution towards such new architectures. Given the ubiquity of the Internet, a clean-slate change of the entire infrastructure to a new architecture is impractical. It is believed that we will see new network architectures coming into existence with support for interoperability between separate architectural islands. We may have servers, and more importantly, content, residing in domains having different architectures. This paper presents COIN, a content-oriented interoperability framework for current and future Internet architectures. We seek to provide seamless connectivity and content accessibility across multiple of these network architectures, including the current Internet. COIN preserves each domain's key architectural features and mechanisms, while allowing flexibility for evolvability and extensibility. We focus on Information-Centric Networks (ICN), the prominent class of Future Internet architectures. COIN avoids expanding domain-specific protocols or namespaces. Instead, it uses an application-layer Object Resolution Service to deliver the right "foreign" names to consumers. COIN uses translation gateways that retain essential interoperability state, leverages encryption for confidentiality, and relies on domain-specific signatures to guarantee provenance and data integrity. Using NDN and MobilityFirst as important candidate solutions of ICN, and IP, we evaluate COIN. Measurements from an implementation of the gateways show that the overhead is manageable and scales well.**

***Index Terms*—Future Internet Architectures; Interoperability; Information-Centric Networking**

I. Introduction

The evolution of the Internet has been driven by demand from new applications and services, the majority of which facilitate improved information delivery. Even though IP is ubiquitous, a number of network research projects have proposed radically new architectural designs to improve delivery of capabilities in a more timely or convenient way, rather than using IP as the common network layer protocol. Each of these solutions revisit and challenge the principles behind IP, primarily in terms of its communication model aspects, such as addressing, connectivity, and mobility support. With the ubiquity of IP and the scale of the Internet, managing the evolution of the network layer and the infrastructure is likely to be increasingly difficult. The *ossification* of the Internet may lead to a "ManyNets" world rather than the "OneNet" Internet we have today [1]. In other words, there may be multiple "islands" coexisting with each other, each being a separate domain having a unique and distinct network architecture [2].

Reachability across domains with different architectures is important and challenging. We believe a pragmatic approach to manage network evolution would be to design an interoperability framework between these different domains, or in other words, "bridging the many islands". This way, we can avoid having to necessarily change all existing designs (including legacy IP networks), while allowing different architectural designs (including the several 'future Internet architectures' being currently considered [3]) to advance and be used in their own domains. In addition, we seek to support the evolution to even other future Internet architectures, thereby sustaining research into such architectures.

There have been many attempts over several decades addressing interoperability across network architectures in the form of multi-protocol routers and gateways. We believe that the concept needs to be revisited today, since compared to previous efforts, the architectures we seek to support for interoperation are much more different in nature than in the past. Some of the proposed architectures for the future Internet challenge the very core idea of the IP-based and host-centric architecture of today's Internet. An important class of architectures that attempt this are Information-Centric Networks (ICN) [4]–[10]. Focusing on *what* rather than *where*, ICN proposes a content-centric, location-independent network layer, motivated by the fact that the main purpose served by today's Internet is information delivery, rather than mere computer-to-computer connectivity [4]. ICN has gained a great deal of interest from both academia and industry in the recent years [11], [12]. ICN has a set of key features and benefits, such as location-independent forwarding [4], content-oriented security [13], and in-network caching [14].

In this paper, we propose COIN, a framework for interoperability between legacy and future Internet architectures, focusing on the important class of ICNs, which are significantly different from today's IP architecture. COIN does not require any change in existing individual domain architectures and preserves their key features and mechanisms. Additionally, COIN does not require content to be moved or replicated to allow access to it from users and end-systems that are in a different network domain. It also does not require the identity/name of a content item to be replicated for each existing domain and end host's understandable semantics. For content-oriented interoperability, it is key that naming is harmonized across different domains. An integral part of this harmonization is that the native naming schema of each domain type (*e.g.*, hierar-

chical structure of NDN [5]) is retained while enabling access and retrieval of content across domains with different naming schema. This goal would be very difficult to achieve with an overlay approach, especially for traversals across multiple domains of different architectures [2]. On the other hand, an efficient translation of content requests and responses between domains can support this interoperability. To this end, rather than creating a new universal layer or overlay, COIN provides translation-based interoperation across multiple domains, with the use of gateways that process requests/responses and retain state information. COIN supports both static and dynamic content requests. Using encryption/decryption as well as iterative signatures performed as we cross from one domain to another domain, COIN ensures confidentiality, integrity and provenance. To obtain content names in a foreign domain, COIN incorporates an Object Resolution Service (ORS) [15]. ORS is an important capability that enables cross-domain name retrieval and usage through a systematic, application-layer procedure. Our ORS design, importantly, relieves the interoperability framework and content providers from renaming content for each domain, and consumers from having to understand new name formats foreign to them. Focusing on the candidate cases of IP, Named Data Network (NDN) [5] and MobilityFirst (MF) [6], we implement and experiment with our design for interoperation. We focus on NDN and MF for two reasons: 1) The Future Internet Architecture (FIA) community treats these as two prominent ICN projects [3], with continuing research efforts and community involvement. 2) The two architectures have significant differences and represent two different "sub-classes" of ICN architectures: NDN supports hierarchical naming with implicit name resolution in the network, while MF supports flat names with explicit name resolution. Taking these differences into account, we evaluate COIN and show that it is effective and efficient.

The contributions of this paper are: 1) a generic interoperability framework among ICN and IP domains for secure static and dynamic content retrieval that preserves key features of domains, thus managing evolution in a flexible manner; 2) an implementation of the framework [16] for interoperability among IP/HTTP, NDN and MF; and 3) measurements based on the implementation of the framework across different domains to demonstrate its utility from a performance perspective.

II. Background and Related Work

A. Information-Centric Networking

Information-Centric Networking (ICN) enables access to named objects, independent of their locations. There have been a number of different ICN proposals in the past decade, *e.g.*, NDN [5], MobilityFirst (MF) [6], DONA [7], XIA [8], NetInf [9], and PURSUIT [10]. In this paper, we mainly focus on two notable ICNs, namely NDN and MF. There are differences between IP and ICNs [4], and also between different ICNs [17]. ICN has several key features and aspects, which we wish to support and preserve while enabling interoperation:

1) *Naming:* In ICN, the network layer is aware of names, while in IP it is only aware of addresses. Different ICNs have different naming schemas: NDN uses human-readable hierarchical names [5], while MF uses 20-byte flat names called GUIDs (Globally Unique Identifiers) [6]. An important service is Name Resolution Service (NRS), which maps names to locations, either implicitly (FIBs in NDN [5]). or explicitly (DNS in IP or GNRS in MF [6]). 2) *Name-based forwarding and routing:* The ICN network layer makes forwarding and routing decisions based on names, which provides benefits such as location-independence and inherent support for mobility [4], [6]. MF forwards both requests and responses based on the source/destination network address (like IP) after late-binding of the name to address, while NDN uses reverse path forwarding (RPF) policy for delivering the response back to the consumer, through Pending Interest Tables (PIT) [5]. 3) *Connectionless transport:* While there has been some work on TCP-like additions to NDN [18] and MF [19], ICN primarily enables content request and retrieval without establishing an end-to-end channel, in contrast to today's HTTP/TCP/IP-based connection-oriented communication channel-based content retrieval [4]. 4) *Content-oriented security:* ICN secures the data itself, as opposed to IP's channel-based and host-based security [13]. NDN uses a trust schema [20] while MF uses self-certifying objects [6] to ensure provenance and integrity. 5) *In-network content caching:* ICNs typically cache content, indexed by names, at every router [14]. This extends the selective and limited CDN-like caching done in today's IP. Many studies have shown ICN caching to be very beneficial for reducing the response time for content delivery as well as availability, especially at the edge [21], [22].

B. ICN Interoperability

There have been different recent approaches for interoperability involving ICNs (surveyed in [23]):

1) Tunneling (Overlay/Underlay): Some approaches use ICN-over-IP tunneling. For example, the basic design of [24] is an NDN-overlay: NDN packets are encapsulated into UDP, TCP or native IP packets traversing IP routers. This enables incremental deployment of ICN over IP and has been used as the starting point for development of software packages of most ICN architectures [4], [9], [10], [25]. Work in [2] introduces a layer 3.5 as overlay, and this layer allows new architectures such as NDN to run. In this approach, each new architecture would have its own layer 3.5 protocol, having to go through the overhead of mapping from/to the underlying layers. Similarly, each overlay has its own naming schema. However, the work does not go into details on how the right name to use is chosen or obtained by a requesting client. IP-over-ICN solutions, such as [18], [26], [27], allow legacy (HTTP/TCP) applications function across an ICN infrastructure, where IP packets are encapsulated in NDN headers, which get decapsulated when leaving the NDN domain. These solutions typically assume a single ICN architecture universally deployed (*e.g.*, NDN [5]) and build IP capabilities on top of it. Also, they deal with added IP-to-ICN (and back) mapping latencies at certain routers on the path [23].

2) Hybrid Approach: An approach to enable evolution to new architectures and interoperation is to add the semantics of a new architecture into an existing one. This results in a new hybrid network layer that is backward-compatible with the native version of the original architecture. CLIP [28] uses an IPv6 subnet prefix for content to enable ICN in IP. Work in [29] proposes the combination of HTTP and ICN, arguing that they both follow a content-centric pattern. Most recently, hICN [30] proposes to encode NDN-specific components into IPv6, and allows the coexistence of IP and ICN dual stacks at hICN-enabled routers (capable of processing both legacy and ICN-enhanced IP packets), while also making use of regular IP routers (capable of processing legacy IP packets). Consumers and data providers, however, still need to have the same semantic understanding, *e.g.*, in terms of naming and how "network" names get mapped to "application" names.

3) Translation: Solutions in [31]–[33] perform direct translation between HTTP and NDN/MF traffic. Translation-based interoperability solutions bring great advantages, such as not having to change domain-specific mechanisms. Work in [31] further optimizes the ability to cache in the network by adding heuristic rules. Moiseenko *et al.* [34] modify NDN packets to better support HTTP-like communication (*e.g.*, uploading large data using POST). These solutions either support only 2 domains (IP plus either NDN or MF), do not support some of the key domain capabilities, and/or require heavy changes to end nodes and routers in existing domains. Our approach overcomes these shortcomings using translation-based stateful gateways for interoperating multiple domains with different architectures, while preserving their key features.

III. Motivation and Overview

A. Design Goals and Rationale

COIN provides interoperability between legacy (current) and future architectures guided by these requirements:

- It should support host-centric (*e.g.*, IP) *and* information-centric (*e.g.*, NDN and MF) networks.
- It should add no architecture/protocol change to the existing individual domains (*i.e.*, no new layer or protocol change).
- Introduce minimal change to end host logic, so clients in one domain use native mechanisms to seamlessly exchange information with another domain.
- Support request for both static (*e.g.*, find a movie) and dynamic content (*e.g.*, query for current weather information), potentially across multiple ($\geq$2) domains.
- Preserve domain-specific features (§II-A); *i.e.*, interoperate between different naming schema, between connection-oriented and connectionless transport, between stateless and stateful forwarding, between channel-based security and content-oriented security, and support in-network caching.
- Each domain's content namespace should be limited only to include the objects in that domains.
- Inter-domain message exchanges must be secure (*i.e.*, provenance, confidentiality, and integrity ensured).

To satisfy the above requirements, we use a translation-based approach primarily to retain each domain independently (and preserve their key features), without changes to existing architectures (thus allowing for easier deployability and evolution). This approach overcomes some of the shortcomings and challenges of alternative approaches:

- Tunneling (overlay/underlay) and hybrid approaches require both the consumer and content provider to have the same semantics and formats, including components such as the naming schema [2], [30]. Translation can achieve the goal of every end host only having to "speak its own language".
- Overlays cannot take advantage of all the capabilities in the underlying domain since the underlay usually does not understand the semantics of the overlay; *e.g.*, in ICN-over-IP [24], IP does not provide many of the advantages that would be obtained from ICN, such as content caching or stateful forwarding. Hybrid approaches are also limited in terms of satisfying all key ICN features through their integrated protocols [23], [30]. With translation, we can retain domain-specific features as well as essential ICN features across domains.
- Overlay approaches introduce considerable overhead and complexity at the overlay-enabled routers, having to perform the mapping between the different decoupled layers; this may be encountered at (potentially) many routers on the path [23]. With translation gateways, routers retain their native domain-specific designs and implementations.
- The addition of new architectures requires significant changes to the overlay at tunnel end-points, both in terms of standardization and deployment. The same is true with hybrid approaches, which requires the embedding of domain-specific components of the new architecture into the integrated network layer protocol. With translation, we can add any number of new network types and attach them to existing domains via gateways supporting them.
- The above challenges become even more severe when dealing with a multitude of interoperating domain types (more than two). We alleviate this in COIN.

It is sometimes noted that translation-based interoperation is counter to the end-to-end principle of the Internet, as argued in [2]. However, we believe that new architectures (mainly ICNs) already challenge the pure end-to-end principle; *e.g.*, in NDN, the procedure for requesting and receiving content is asynchronous, with routers managing transport on a hop-by-hop basis, without necessarily having a complete end-to-end communication [5]. Also, in today's Internet, middleboxes such as NATs add additional indirection in the network [35]. We believe a translation-based approach is suitable and pragmatic for interoperability between current and future networks.

With a focus on content-oriented services, our translation is performed at the "content name level", *i.e.*, in the layer that identifies content names, be it the application layer in legacy IP domains (*e.g.*, URLs to identify content in HTTP) or network layer in NDN domains (in form of hierarchical human-readable names). This provides a significantly higher abstraction than the address-based design of legacy interoperation and is important since names are "first-class" entities in information-centric paradigms (§II-A). In such environments,

it is also important for consumers to pick "the right name" for a content request, and receive that content. Recent works such as [2], [30] allude to the importance and challenges of such mechanisms, although they do not provide a solution for it. We propose a protocol for Object Resolution, to enable the retrieval of the necessary names (which we explain in §IV-G).

B. Overview of COIN

COIN provides interoperability among any number of domains, each having a distinct network design and architecture, including legacy (IP) or future (ICN) Internet architectures. COIN gateways provide this interoperability through translation and state maintenance. A client in one domain can request for content (static or dynamic) multiple domains away, and receive the corresponding content in the response. COIN makes no change to existing domain-specific architectures, and preserves key domain features, including domain-specific security models and mechanisms. Most notably, COIN preserves namespace size and structure of each domain, and does not create a new naming schema. Content can be universally identified using its domain-specific (*i.e.*, native) name, plus its domain ID. A client requesting content from another domain, uses the content's native name and its domain ID. To acquire that information, COIN uses an *Object Resolution Service (ORS)*, which is an application-layer search engine-like service providing names as response to keyword queries. The foreign name provided to consumers are not distinguishable from native ones, thus making the consumer's request for content seamless.

C. Addressing Challenges for Gateway-based Interoperability

While our solution (overviewed in §III-B) helps achieve our design goals, there are additional concerns to be addressed. Most of these challenges exist for other translation-based approaches as well. We explain how COIN overcomes them.

Evolution flexibility; Too many pair-wise translators? Typically, in a translation-based interoperability solution, for n different network architectural designs, one might end up needing n^2 translators [2]. Not only would it be too complex to design so many translators, it also can make it very inflexible for adding a new domain architecture: n additional translators would need to be implemented. COIN overcomes this by having an *internal canonical form* at the gateways, and *adapters* that convert domain-specific packets to/from this canonical form (explained in §IV-F). This way, for n different network designs, we will only have n adapters at the gateway (rather than n^2 individual translators), and one canonical form that is consistent across all gateways. Note that this canonical form is not 'yet another network layer'; it is only an internal design component inside the gateway.

Too many requests going through gateways? Only the requests going across a domain to another domain need to go through a gateway. However, this may still end up resulting in an excessive number of requests that a gateway has to process. This can make the gateway un-responsive and be a single point of failure. This is a general problem of gateway-based

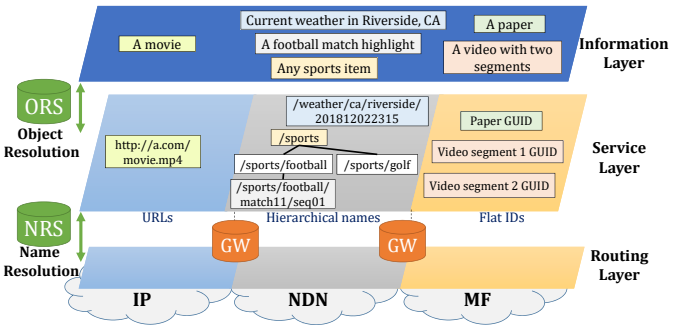


Fig. 1. Layered architecture overview

interoperation. To overcome this, COIN leverages in-network caching, a key domain-specific feature that COIN preserves, because of its use of the native naming schema of the domain. Content coming from another domain through a gateway can still be cached in the consumer domain (§IV-F). Many works have shown that in-network caching is very beneficial since the content demand in the Internet follows a Zipfian distribution [21]. Work in [22] has shown that with a proper caching scheme, in-network caching for a typical web workload can achieve up to 70% hit rate even with cache capacities as little as 2% to 7% percent of the whole content space. With caching enabled, and assuming Zipfian workload, the majority of the requests in COIN would be satisfied in the consumer domain, thus not having to necessarily go to gateway to be processed.

Storage overhead at gateways? Gateways have to keep a small amount of information as state for every incoming request. While in-network caching can dramatically reduce the number of requests the gateway has to process, storing state associated with each of them may still be a challenge. To overcome this, gateways in COIN leverage *request aggregation* for requests for the same content (typically static content) (more details in §IV-F). Only the first request is forwarded, while the subsequent identical ones get aggregated (similar to NDN PIT [5]). In addition, COIN gateways can cache content themselves, thus enabling them to respond without having to issue a new request (and thus store associated state) in the next domain. These methods, combined, greatly reduce the amount of memory consumption at the gateways as well as the number of requests going out of them.

IV. Architecture and Design

A. Preliminaries

COIN's network environment may be made up of a number of different domains (*e.g.*, IP, NDN, and MF) with gateways connecting pairs of domains, in addition to clients, servers, publishers, and content repositories that can reside in any of the domains. A high-level, layered architecture view is shown in Fig. 1. We identify three layers (similar to [36]), each characterizing an important aspect of COIN's content-centric view. The *Information layer* captures accessible objects and content items in various applications. A *Service layer* shows in what format (hierarchical format, *etc.*) each object in an Information layer is named and identified. The *Routing layer*

takes care of transmitting packets to an appropriate (one or more) recipient(s) using routing/forwarding protocols within that domain. It is important to note that we are not adding any new layers; rather, we are recognizing the logical layers that represent functionality that is common in the domain-specific architectures. For example, the service layer is part of the network layer in NDN and MF, and part of the application layer in IP. The Object Resolution Service (ORS) generates names understandable by the corresponding domain's service layer and the domain-specific Name Resolution Service (*e.g.*, DNS in IP and GNRS in MF). It helps names to be mapped to location information in each domain for routing. The figure also shows that each domain (and producers and consumers in the domain) only needs to understand its own naming structures (whether hierarchical or flat). Gateways facilitate appropriate translations and bridging between different network architectures, preserving their key features and internal mechanisms.

B. Service Interface

The primary services supported by COIN are static and dynamic content retrieval. Both follow a query/response model, a very popular model in today's Internet as well as in ICNs [4]. This distinction between static and dynamic content requests is important, since they need to be treated differently: The response for a dynamic content request might depend both on consumer's input and the current state of the server (such as a keyword-based search that may depend on time, location, or server policies). Thus, the response cannot be from a cache, since the current server-generated response is desired. Static content requests, on the other hand, have no such restriction. Cached content, in routers or Content Delivery Networks (CDNs), can be returned to consumers, as long as it has the right version. Note that other possible services, *e.g.*, publish/subscribe [37] are not the focus of COIN's current design. However, with additional modules for processing those other services, *e.g.*, push-based multicast or repetitive poll-based request generation component implementation for publish/subscribe with the capability to translate to/from an internal canonical format, COIN can support them as well.

C. Common Information Elements

These are common elements on which the translation between different formats have to be performed and are primary parts of a COIN gateway's canonical form. COIN supports all protocols that have these common elements (*e.g.*, HTTP/IP, NDN, MF, XIA, NetInf, and even FTP):

- *Request type* (to distinguish between dynamic and static content requests; request type can be a pre-existing field in the packet header or body based on implementation choice);
- *Destination domain and content name* (generally as "DstDomain/ContentName" to identify content and target domain);
- *Content version* (as content may have different versions);
- *Exclude for static content request* (to allow consumers to get the latest version of a content item);
- *Input for dynamic data request* (to allow a consumer to pass parameters to a dynamic data provider); and,
- *Demultiplexing key* (to identify a corresponding request when the response data comes back to a gateway).

D. Naming

Naming is key to enabling content-oriented interoperability. COIN primarily performs translations at name level. This principle brings important benefits: 1) Each domain keeps its naming schema (*e.g.*, NDN's hierarchical naming [5] or MF's flat IDs [6]), which helps with evolvability. While consumers have to use a globally unique expression for each content as "ContentDomain+ContentName", they do not have to understand such a syntax. It will look seamless to users, as if they are using a native name. This is facilitated by the ORS, as described in §IV-G. For example, a consumer in an IP domain requesting for content named "/ICCCN/papers/COIN.pdf" in an NDN repository, will send an HTTP request for "http://NDN/ICCCN/papers/COIN.pdf", which is just like any other HTTP request. After going through gateway processing, the NDN repository receives the request as an NDN Interest with name "/ICCCN/papers/COIN.pdf", just like any other NDN Interest. 2) Each domain keeps its namespace size, which helps with scalability. No domain has to keep track of, or maintain, another domain's content namespace (just needs the IDs of other domains). 3) The name-to-location mapping in each domain, utilized in order to deliver to/from gateway, consumer, or producer, is handled by the domain's already existing name resolution service (*e.g.*, GNRS in MF).

E. Transport, Routing and Forwarding

Motivated by its design principle, COIN allows the composition of connection-oriented and connectionless transport across domains. For example, if a consumer using HTTP/TCP/IP is requesting content from a producer in NDN, the gateway acts as the second end of the TCP connection (*i.e.*, similar to a proxy server listening on an HTTP port) to the consumer (establishing sessions, *etc.*), while acting as a typical NDN client (sending a content Interest in a connectionless manner) towards the NDN side. When the data comes back, the gateway sends the data to the consumer, using the stored IP address and source port information of the consumer.

Similar to today's Internet, COIN decouples intra- and inter-domain routing [2]. Domain-specific routing mechanisms can be leveraged, and be stitched at gateways. In the case of multiple gateways between two domains, inter-domain routing can be used to connect one gateway to the nearest other gateway. Existing architectures need not know or implement these inter-domain algorithms. As for state, gateways connecting the same domains on either side can exchange and share state information, so any gateway can process the response.

COIN gateways process requests and responses differently, with an important distinction being that response forwarding is stateful (§IV-F). Importantly, gateway forwarding conforms to domain-specific forwarding policies; *e.g.*, NDN has a reverse path forwarding (RPF) policy where every node traversed by the request has to be in the responses path [5]. Other architectures, such as MF, may not have such a restriction, thus

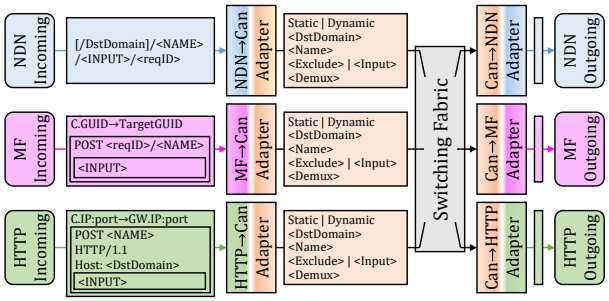


Fig. 2. COIN gateway design: processing requests

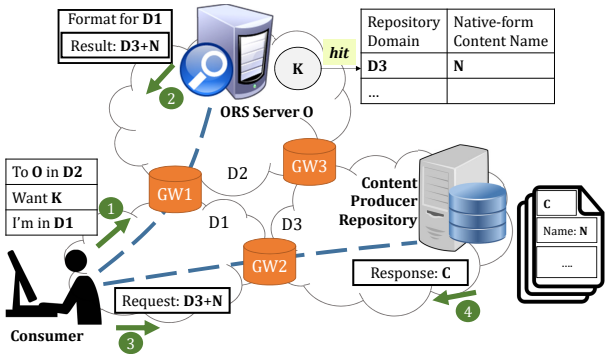


Fig. 3. A schematic view of object resolution and content retrieval in COIN

allowing secondary gateways to route the response back towards the consumer. Conforming to such forwarding policies, COIN can provide a seamless interoperation across domains, without having to change existing infrastructures.

F. COIN Gateways

A COIN gateway translates requests for information received from one domain to a request meaningful in the adjacent domain (and similarly for responses). We design and implement the interface to each distinct domain as a "pluggable adapter" on the gateway in each direction. We choose to translate the incoming request or the headers of the response to an "internal" canonical form (IV-C).

Incoming request processing involves recognizing whether the request is for static or dynamic content. For NDN, a request with a specific version is seen as a request for dynamic content while a request with just a prefix (and exclude) is for static content. For HTTP, we use POST and GET methods as dynamic and static content respectively. It also determines the destination domain based on the "Host" field in case of HTTP, the destination GUID in MF, and the domain prefix in NDN. The opaque string (from the originating domain's perspective) that is the name on the destination domain will be extracted from the request (marked as the field "$<Name>$" in Fig.2. For dynamic requests, the incoming request processing recognizes the body of the POST in MF and HTTP, and the penultimate component of NDN name, as the request input. The demultiplexing entity ("$<Demux>$") depends on the different cases. For static content requests, we use the tuple $<domainName, contentName, exclude>$. For dynamic content requests, we use client $<IPaddress, port>$ (socket) for HTTP case, client $<GUID, reqID>$ in MF case and $<clientID, reqID>$ in NDN case.

The incoming request processing results in an internal canonical request (orange boxes in Fig. 2). The gateway can respond to requests for static content from the local cache, aggregate requests for the same static content (with same exclude) or consume them. The remaining requests (in canonical form) are sent to the "switching fabric", where inter-domain routing determines the forwarding to the proper outgoing request processor. The outgoing request processing forms a domain-specific outgoing request.

When the response returns, the gateway matches it based on the state information and forwards the content to all the pending requests waiting on this key (similar to matching a PIT entry in NDN). This enables native multicast, similar to NDN. We use the "Last-modified" field in HTTP and MF and the version field in the NDN name as the version of the response. The gateway sends the version using the domain-specific format.

G. Object Resolution

In COIN, an important step in requesting a piece of content residing in another domain is to acquire the content's name and the ID of the domain it is in. This is achieved using an Object Resolution Service (ORS). ORS is an application-layer search engine that: 1) returns names for keyword queries, and 2) leverages a combination of crawling, registration and indexing methods to gain and store knowledge of content names. This way, ORS plays the same role that today's popular search engines, *e.g.*, Google or Bing do. More specifically, today's single-domain search engines could be considered as a special case of ORS. ORS has to additionally take both the content's and consumer's domain(s) into account: it has to provide the content domain ID in its query result, presented in a format understandable in the consumer domain (and thereby by the consumer). This is an important design choice, as having ORS servers that understand multiple domain languages avoids having all data providers/servers in the world learn the other domains' languages. There has been prior work on ORS in ORICE [15], which we use and extend in COIN.

Fig. 3 shows a high-level schematic view of the object resolution procedure. There are three domains with different network architectures $D1$, $D2$, and $D3$, with three COIN gateways stitching them together. The consumer, object resolution server O, and content repository for content C reside in $D1$, $D2$, and $D3$ respectively. We put the consumer and the ORS server O in two different domains to show a more complicated scenario; normally, $D1$ could have an ORS server too which the consumer can ask without having to go across domains. The consumer generates a query for keyword (phrase) K, asking ORS O in $D2$. Identifying the ORS's name and its domain are important to make sure that the query goes to the right gateway. Although the figure only shows the information at a high level, the specific formats depend on what the domain is. For example, if the consumer is in an IP domain, he will perform a DNS lookup on "D2", obtaining the IP address of $GW1$. In NDN, the consumer will send a packet with prefix "/D2/O/", which will be directed to $GW1$ ($GW1$ has already

announced and registered itself as "D2" in $D1$). The consumer also specifies that he is in $D1$; so O generates the result in a format understandable to a user in $D1$. With the help of $GW1$'s translation and state-maintenance, the query can reach O and its result sent back to the consumer. Some packet-level details, such as demultiplexing keys are omitted in the figure. More on protocol exchange details are in §IV-H (ORS query is an example of a dynamic content retrieval).

Upon receiving the consumer's query, O searches for K in its database of indexed content names and their domains (including content in $D3$). The way this database is managed is similar to today's search engines' crawling and registration methods; more details are provided in [15]. Assume K hits one entry with a content named N in domain $D3$ (for presentation simplicity, we assume only one item in the result, while in practice there can be many more). O generates a result combining $D3$ and N ("D3+N" in Fig. 3), formatted for $D1$. What N looks like depends on the naming structure of $D3$; *but* the formatting of the result depends on the semantics of $D1$. For example, if both $D1$ and $D3$ are in an HTTP/IP domain (as in today's Google search), then N would be a URL (*e.g.*, "abc.com/def"), formatted and presented to the consumer as "http://abc.com/def" (no indication of $D3$ is needed for same-domain pairs). As another example, if $D3$ is MF, then N would be a content GUID (*e.g.*, "1234"). If $D1$ is NDN, then the result would be the enriched name "/MF/1234". More combinations are described in [38].

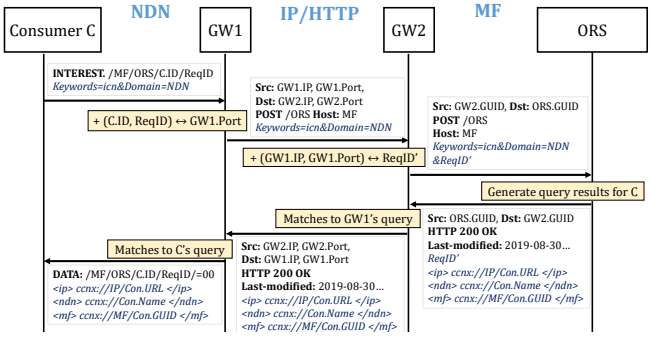
After gathering the result, the consumer generates a request for the content itself, using the acquired name N combined with $D3$, getting routed to $GW2$. Note that for this purpose, the consumer's own domain ID is not needed, since the repository returns content C (named N), not knowing (and not needing to know) where the consumer resides.

While at first glance, it may seem a burdensome task to acquire names through the ORS for content requests, it follows the pattern that users use on the Internet today in practice [15]. For example, most often, retrieving a webpage for the music video of the 2017 song "Despacito", is proceeded by a (Google) search such as for "Despacito music video". ORS in COIN plays the same essential role as the search engine. It is also worth mentioning that ORS is a service which can be provided by many entities (as we have Google, Bing, *etc.*) and each can have many physical servers. We believe ORS is an important, convenient service to deploy by ISPs or third-party entities, and provides benefits for interoperability.

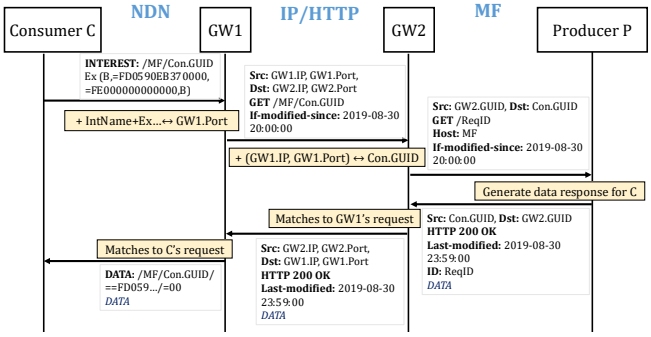
H. Protocol Exchange

To illustrate the translation-based exchange for interoperating across multiple domains, we use a 3-domain setting where a consumer residing in an NDN domain wishes to receive content from a server/producer residing in an MF domain, with an IP domain in the middle (Fig. 4). We examine two cases: dynamic content retrieval (DCR) using the example of ORS (Fig. 4(a)); and static content retrieval (SCR, Fig. 4(b)).

The three different architectures take care of content naming at different domain-specific layers: the HTTP application layer,



(a) Dynamic content retrieval (object resolution example)



(b) Static content retrieval

Fig. 4. Protocol exchange across 3 domains

in IP; network layer in NDN; and either HTTP or network layer in MF. When a client generates an NDN Interest, to enable correct translation, we use the destination domain ID in the name. To distinguish between DCR and SCR, we use POST and GET methods in HTTP respectively; we check the existence of Exclude or Input in NDN Interests. For DCR (Fig. 4(a)), the retrieved response should not be from a cache, since the current server-generated response is desired. This requires individual requests to be distinguishable (globally unique), to have the correct response-to-request mapping at the servers and gateways, including even those made by the same client. In TCP/IP, client IP and port numbers provide this demultiplexing capability. For NDN and MF, we introduce a unique Request ID (ReqID) generated by the consumer or gateway. The ReqID can be a component of the DCR Interest name in NDN, and part of the request payload in MF. Gateways create state (marked as '+' in the Fig.) associated with each outgoing request, and maintain state for demultiplexing. For example, in Fig. 4(a), the mapping on an NDN-to-IP gateway is a 3-tuple of <Client ID, Request ID, GW1 port number>. When the response data is returned, the gateway can find the corresponding request based on its port number (which is the source port number that was previously used to connect to GW2) in the response. As can be seen in Fig .4, using domain-specific naming, in-network caching can be supported and provide benefits, in COIN. Although we show 3-domain examples here, details and protocol exchange scenarios for all possible 2-domain cases are provided in [38].

I. Security

Securely bridging communication across different network architectures that have different security models and mecha-

nisms seamlessly, without significant changes to the individual architectures is challenging. We aim to unify different security models across the architectures. They may be classified as being either channel-based (for host-centric networking *e.g.*, IP), or content-oriented (for ICNs). The fundamental distinction between the two security models lies in the relationship between the "name" layer (content retrieval functionality) and security layer (ensuring confidentiality, provenance, integrity, *etc.*) functionality in the service layer (§IV-A). With connection-oriented security the security layer (TLS/SSL) operates below the name layer (HTTP). Interoperability gateways do not have access to information such as keys, as they are encrypted. For interoperability, the gateways have to decrypt the information exchanged to get the name and other features required for content retrieval. In contrast, content-oriented security may just encrypt the data (payload) and leave the content-retrieval headers (*e.g.*, NDN/MF headers, including content names) in the clear. Thus, gateways can reformat the headers without modifying or having to access the payload. COIN supports a number of mechanisms to unify access to information across these two security models. We focus on two important security use cases of COIN: Encryption (to ensure *confidentiality*); and Signatures (to ensure *provenance* and *integrity*). The mechanisms presented here are security-enhancements to protocol exchange presented in §IV-H.

1) Encryption: Encryption prevents unauthorized network nodes (including eavesdroppers) from accessing confidential content. The common approach to achieve this is to encrypt the data (*e.g.*, RSA and ECC [39]) or encrypt the channel between the data consumer and producer (*e.g.*, HTTPS). The producer and a (set of) predefined (authorized) consumer(s) have to agree on a common encryption mechanism. We focus on content retrieval across compositions of content-oriented (ICN) and channel-based (IP) security models, with endpoints having the same security model, and when they are different.

Case 1: Both endpoints with content-oriented security model, and intervening domains with channel-based security. We consider a scenario with a consumer and producer in two separate NDN domains using content-oriented security, and an IP domain in between using channel-based security. Fig. 5 shows COIN 's encryption-enhanced protocol exchange for this case. With both the consumer and producer using content-oriented encryption, the authorization information ($auth_C$) and $Data$ would be encrypted when traversing the gateways. The authorization information can be the consumer's public key (pub_C), following a priori consumer-producer consensus on the authorization mechanism. The gateways simply translate between NDN names and HTTP/HTTPS URLs, without needing to decrypt and/or re-encrypt $auth_C$ or $Data$. Thus, COIN ensures end-to-end confidentiality.

Case 2: Either endpoint with channel-based security When at least one of the two endpoints (consumer and/or producer) uses channel-based security (*e.g.*, HTTPS) and the other(s) use content-based security, gateways would then need to re-encrypt the data retrieved in one domain to provide confidentiality while delivering the content to the other domain.

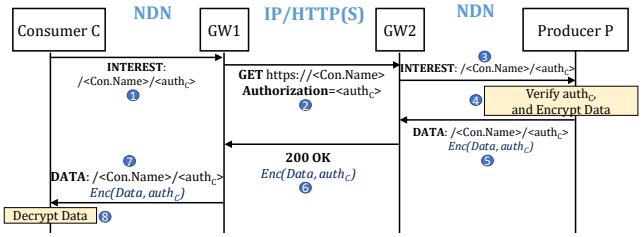


Fig. 5. NDN/IP/NDN encryption

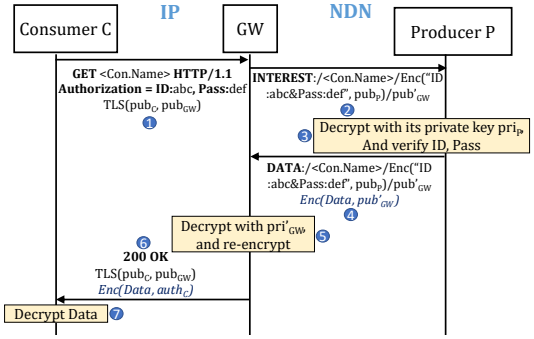


Fig. 6. IP/NDN encryption

COIN's gateways borrow ideas from the popular state-of-the-art solution of HTTPS proxies. The gateway has to decrypt the HTTP header inside a TLS connection, in order to discover the content name (URL). The gateway acts as a proxy, trusted by both consumer and producer. We believe this is acceptable, as it is a well-established practice to trust HTTPS proxies.

Unlike Case 1, COIN 's mechanism in Case 2 decouples encryption and authorization, to allow composition of two different security models of the end points: An HTTP/IP end point achieves this by using the HTTP header "Authorization" field, or Web-based authorization (the consumer provides the username and password, which are carried in the HTTP request body). The producer can then verify the authorization.

Fig. 6 shows an example for this case with a gateway between HTTPS/IP and NDN. The consumer in the IP domain requests the content as if the gateway is an HTTPS proxy, by establishing a secure connection to the gateway using the public key of its own (pub_C) and the gateway (pub_{GW}) (Diffie-Hellman key exchange in TLS). In the HTTP header (or body) over the TLS connection, the consumer sends the content name and the authorization information (username and password in step 1 in the Fig., or alternately the public key of the consumer). The gateway creates an Interest in the NDN domain. We make minor modifications to the name in NDN (to provide the authorization), using the format "/$name$/$auth_C$/$encrypt$", to provide the producer with the needed information for authorization and encryption. In the Fig., the gateway encrypts the authorization information with the public key of the producer (pub_P) and uses its own public key (pub'_{GW}). The gateway could use different key pairs (*e.g.*, pub'_{GW} and pub_{GW}) for the two different domains (step 2). For MF, the request packet format would include a new field for the authorization information. When the field is not set, authorization information is used as encryption information, as in Case 1. On receiving the request, the producer will

decrypt it using its own private key and verify the authorization information (step 3). Upon verification, the producer sends the NDN Data packet (to the gateway) whose payload is $Data$, encrypted by pub'_{GW} (step 4). The gateway decrypts the data with its own private key pri'_{GW} (step 5) and sends the data over the TLS connection to the consumer (step 6). The consumer can then decrypt and access the data (step 7). The reverse, ICN-IP scenario, would follow a similar pattern.

2) Signatures: In the scenario where the producer allows the content to be shared with *anyone* in the network. The consumers need to verify that: 1) the data is coming from a trusted producer (*provenance*) and 2) no one on the path has tampered with the content (*integrity*). To ensure the integrity of the content, a cryptographic hash function (*e.g.*, MD5, SHA-1, SHA-256) can be applied to the data and announced to the consumer. Provenance is verified by a digital signature: the hash encrypted by the private key of the producer (*e.g.*, RSA signing, ECDSA, EdDSA [40]). The consumer can decrypt the signature with the public key of the producer, and compare the result with the hash of the content, possibly followed by some trust schema [20]. For interoperability across different domains, it is highly likely that the consumer may not understand the producer's signature algorithm or the trust schema (or both). To overcome this, COIN takes advantage of *transitive trust* [41] with domain-by-domain signatures: the gateway on the producer side verifies the provenance and integrity of the data on behalf of the consumer and re-signs data with its own private key for the next domain. The consumer verifies (and trusts) the last hop gateway.

Fig. 7 shows an example of our solution spanning 3 domains. After receiving the request, the producer will sign the data (D) with its own private key (pri_P) based on the signature algorithm in the domain (SA_{MF}). On receiving the content, $GW2$ will verify the provenance and integrity using the public key of the producer (pub_P) on behalf of the consumer, since it understands the signature algorithm and can also utilize local certificate authorities (CAs) to check its trustworthiness. Once $GW2$ confirms that the content is trustworthy, it will re-sign the data with its own private key (pri_{GW2}) using the signature algorithm in the IP domain. $GW1$ will thus trust the producer, since it trusts $GW2$ (due to transitive trust). Once the signature is verified using $GW2$'s public key, $GW1$ will forward the data to the NDN domain and sign the content using its own private key. Since the consumer trusts $GW1$, it concludes that the content is trustworthy.

3) Denial of Service (DoS) Attacks: While request aggregation and content caching alleviate COIN from some negative impacts of excessive request and response processing load (that is *benign*) on gateways, *malicious* DoS (and DDoS) attacks can impact gateways' availability. IP and ICN domains, with different security models, can be the source of different DoS attacks, such as bandwidth depletion and reflection attacks in legacy IP domains [42]. With ICN's content-oriented security models, DoS attacks manifest themselves mainly as Interest flooding (too many malicious requests for non-existing content) and content/cache poisoning (responding with fake

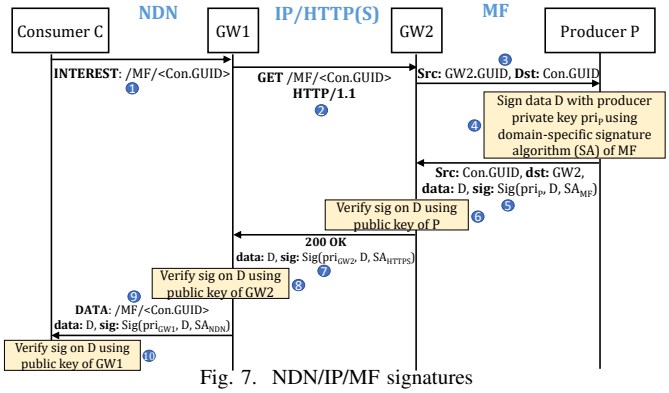


Fig. 7. NDN/IP/MF signatures

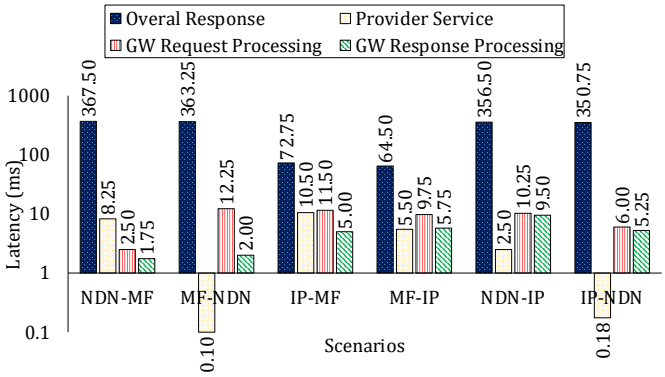
or corrupted content) [43]. In COIN, each domain retains its own security model and mechanisms, to allow development of countermeasures for attacks meaningful in its own domain. Thus, in COIN, DoS attacks in a domain are contained within that domain. For example, with Interest flooding, excessive requests will be dropped at or before the ICN-border gateway (through mechanisms such as statistic-based rate limiting [43], [44]). Similarly for content poisoning, with the proper use of content-oriented signature validations, fake or corrupted content will be detected and discarded at or before the first gateways it encounters.

V. Evaluation

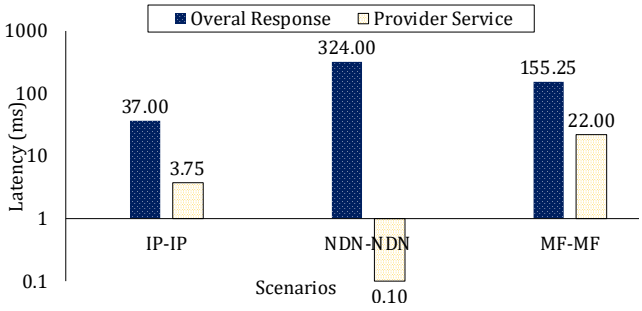
For evaluation, we use a representative implementation for each domain: CCNx v0.8.0 (it contains all the essential components of NDN needed for our framework); the latest version of MobilityFirst project [45] for MF domain; and a basic Linux implementation of IP forwarding. The implementation of COIN, including all its essential components such as gateway and adapter modules are available in [16] as proof of concept. We experimented on various combinations and settings, and observed COIN's ability to satisfy its design goals (§III-A), especially its ability to preserve each domain's key features (as explained in more detail in §IV-I3). While there are a number of aspects to consider including correctness, user convenience, deployment flexibility etc., we focus on the performance of COIN here (we have proved the correctness of the proposed translation procedures in [46]).

A. Forwarding Efficiency

To evaluate the forwarding efficiency of the implementation, we set up a testbed with five VMs with the topology as "$C \leftrightarrow R_1 \leftrightarrow GW \leftrightarrow R_2 \leftrightarrow P$", in which client C and router R_1 are in domain D_1, and content provider P and router R_2 are in domain D_2. Node GW, an implementation of COIN gateway, is in between the two domains and performs translation. Each VM has 1GB memory and runs Ubuntu 14.04. With cases of domains D_1 and D_2 being both distinct (*i.e.*, *interoperation* scenarios) and same (*i.e.*, *native* scenarios), we evaluated all 9 combinations (each being IP, NDN, MF). In native scenarios, GW is replaced by a regular router, with the same configuration as R_1 and R_2. We tested functionality with a



(a) Interoperation scenarios (different domain types for D_1 and D_2)



(b) Native scenarios (same domain types for D_1 and D_2)

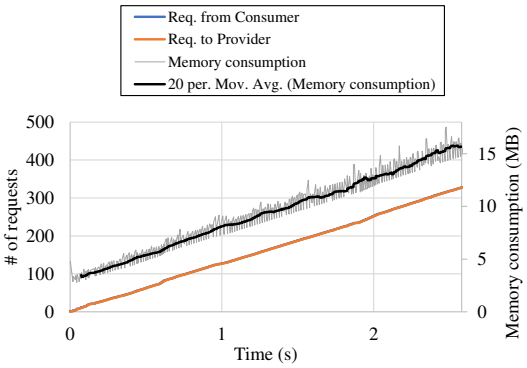
Fig. 8. Latencies (static content retrieval): total response, content provider, gateway request and response processing (logarithmic axes). Note that there are no gateways (and thus gateway latencies) in native scenarios (b).

client asking for content residing in a remote domain of a potentially different architecture, and getting the content back.

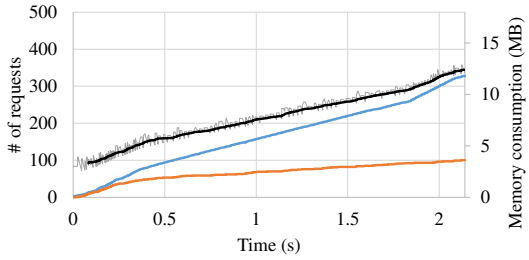
Fig. 8 shows the latencies measured for requests for static content. The overall content retrieval time (response time) at the consumer, the provider's service time and gateway processing time for request and response (averaged over several runs, discarding outliers), are shown. Note the difference in the y-axes of the different bars in the figure. As shown in Fig. 8(a), the processing time at the gateway in interoperation scenarios, including reformatting and maintaining state between the two domains, while not negligible, is reasonable for an initial software implementation. The gateway contributes between 4-19 ms of processing delay, compared to the total response time of 60-360 ms, in this small-scale topology. The gateway contributes a relatively small portion of the overall response time, especially in the ICN cases. It should be noted that the higher response time observed whenever one side is NDN, is not due to the interoperability gateway, but rather the NDN logic itself: the client waits for sending a second query to ensure it has received the latest piece of content. In fact, we observed a response time of ∼300 ms for NDN→NDN on our testbed. This is seen in Fig. 8(b) as well, which shows the latencies for native scenarios, *i.e.*, those with same domain types throughout the topology, with no COIN gateways. Our results show that the overhead of COIN's forwarding is reasonably small.

B. Scalability

We use the ORBIT testbed [47] to evaluate the scalability of COIN. ORBIT is a network of 400 nodes in a grid topology.



(a) Dynamic content retrieval



(b) Static content retrieval

Fig. 9. Scalability: memory consumption and number of requests

Each machine has 4GB memory and runs Ubuntu 14.04. We run each router (forwarding engine), provider, consumer and gateway on separate physical nodes. In our topology, we included 50 consumers (in one domain), 1 provider (in another domain) and 1 gateway. The consumers are connected to the gateway via a pre-configured access network. Our server stalls the response to each request for 3 seconds to batch more requests on the gateway (especially for NDN, and 3 sec. because the request timeout time in NDN is ∼4 sec.)

We measure the amount of state stored in the gateway (memory consumption) *vs.* different numbers of requests from consumers. The implementation is in Java, which has automated memory management. We call garbage collection very frequently during run-time to get a better estimate of memory consumption. This would make our gateway slightly slower compared to production use. We evaluate the requests to static and dynamic content separately. Only the results of the experiments for IP→NDN is shown in Fig. 9.

Evaluation for dynamic content: We have 50 clients sending 328 dynamic content requests simultaneously. Fig. 9(a) shows the instantaneous memory consumption (and moving average over 20 values) *vs.* the number of incoming and outgoing requests for this scenario.

Since consumers request dynamic content, we do not see any aggregation at the gateway – each request from the client results in a distinct request to the provider. Therefore, the incoming and outgoing request values are very close to each other in the Fig. We observe that the memory consumption grows linearly with the number of incoming requests since we keep the states for each request.

Evaluation for static content: Clients make 328 requests spread across 100 static content items simultaneously. The

popularity of the content follows a Zipf distribution with α=0.81. [48] shows that this is a realistic content demand model. The server still waits 3 seconds before sending the response to allow request accumulation. Fig. 9(b) shows the results. Since we keep the state on the gateway, we can aggregate multiple requests for the same content (name). Therefore, the number of requests arriving at the provider is smaller than the number of requests generated by the consumers. The memory consumption grows sub-linearly relative to the incoming requests. The memory consumption is also lower, compared to that of dynamic content for the same number of requests.

We ran the same experiments in other domain combinations (NDN→IP, MF→NDN, *etc.*) and saw similar results. Although keeping per-session state puts additional burden on the gateways (state grows with number of flows), it is analogous (and no worse than) maintaining PIT state at an NDN router. Since COIN allows request aggregation and content caching at the gateways, this interoperability framework scales better.

VI. Conclusion

This paper proposes COIN, a content-oriented interoperability solution as a pragmatic approach to manage evolution towards future Internet architectures. COIN does not change existing architectures (of IP and different ICNs), preservers and uses their key features, enables their co-existence, and is flexible for extensibility and evolvabaility. Through various scenarios and experiments, COIN was shown to make essential content-oriented services (static and dynamic content retrieval) available to consumers across multiple domains, with reasonable efficiency. While we acknowledge that there are many other scenarios and services to consider, especially in the Internet context, we believe COIN is a good starting framework for managing seamless interoperability among multiple domain types of future Internet architectures.

VII. Acknowledgements.

This work was supported by the US Department of Commerce, National Institute of Standards and Technology (award 70NANB17H188) and US National Science Foundation grants CNS-1455815 and CNS-1818971.

References

[1] M. Ammar, "Ex uno pluria: The service-infrastructure cycle, ossification, and the fragmentation of the internet," *SIGCOMM CCR*, 2018.
[2] J. McCauley *et al.*, "Enabling a permanent revolution in internet architecture," in *SIGCOMM*, 2019.
[3] "NSF Future Internet Architecture Project," http://www.nets-fia.net/.
[4] V. Jacobson *et al.*, "Networking Named Content," in *CoNEXT*, 2009.
[5] L. Zhang *et al.*, "Named data networking," *ACM SIGCOMM CCR*, vol. 44, no. 3, 2014.
[6] D. Raychaudhuri *et al.*, "Mobilityfirst: A robust and trustworthy mobility-centric architecture for the future internet," *SIGMOBILE*, 2012.
[7] T. Koponen *et al.*, "A data-oriented (and beyond) network architecture," in *SIGCOMM*, 2007.
[8] D. Naylor *et al.*, "XIA: Architecting A More Trustworthy and Evolvable Internet," *SIGCOMM CCR*, 2014.
[9] C. Dannewitz *et al.*, "Network of information (netinf)–an information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, 2013.
[10] N. Fotiou *et al.*, "Developing Information Networking Further: from PSIRP to PURSUIT," in *BROADNETS*, 2012.
[11] R. Ravindran *et al.*, "5g-icn: Delivering icn services over 5g using network slicing," *IEEE Communications Magazine*, vol. 55, no. 5, 2017.
[12] S. O. Amin *et al.*, "Leveraging icn for secure content distribution in ip networks," in *MM*, 2016.
[13] R. Tourani *et al.*, "Security, privacy, and access control in information-centric networking: A survey," *IEEE communications surveys & tutorials*, vol. 20, no. 1, 2017.
[14] G. Zhang *et al.*, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, 2013.
[15] S. S. Adhatarao *et al.*, "ORICE: An Architecture for Object Resolution Services in Information-Centric Environment," in *LANMAN*, 2015.
[16] "COIN," https://github.com/SAIDProtocol/ICNInteroperability.
[17] S. S. Adhatarao *et al.*, "Comparison of Naming Schema in ICN," in *LANMAN*, 2016.
[18] I. Moiseenko and D. Oran, "TCP/ICN: Carrying TCP over Content Centric and Named Data Networks," in *ICN*, 2016.
[19] K. Su *et al.*, "Mftp: A clean-slate transport protocol for the information centric mobilityfirst network," in *ICN*, 2015.
[20] Y. Yu *et al.*, "Schematizing trust in named data networking," in *ICN*, 2015.
[21] S. K. Fayazbakhsh *et al.*, "Less pain, most of the gain: Incrementally deployable icn," *ACM SIGCOMM CCR*, vol. 43, no. 4, 2013.
[22] S. Li *et al.*, "Popularity-driven content caching," in *INFOCOM*, 2016.
[23] M. Conti *et al.*, "The road ahead for networking: A survey on icn-ip coexistence solutions," *arXiv preprint arXiv:1903.07446*, 2019.
[24] W. Shang *et al.*, "NDN.JS: A JavaScript Client Library for Named Data Networking," in *NOMEN*, 2013.
[25] J. Chen *et al.*, "Coexist: Integrating Content Oriented Publish/Subscribe Systems with IP," in *ANCS*, 2012.
[26] D. Trossen *et al.*, "IP over ICN – The better IP?" in *EuCNC*, 2015.
[27] S. Shannigrahi *et al.*, "Bridging the icn deployment gap with ipoc: An ip-over-icn protocol for 5g networks," in *NEAT*, 2018.
[28] L. Heath *et al.*, "Clip: Content labeling in ipv6, a layer 3 protocol for information centric networking," in *ICC*, 2013.
[29] L. Popa *et al.*, "Http as the narrow waist of the future internet," in *Hotnets*, 2010.
[30] G. Carofiglio *et al.*, "Enabling icn in the internet protocol: Analysis and evaluation of the hybrid-icn architecture," in *ICN*, 2019.
[31] S. Wang *et al.*, "On Adapting HTTP Protocol to Content Centric Networking," in *CFI*, 2012.
[32] F. Bronzino *et al.*, "In-Network Compute Extensions for Rate-Adaptive Content Delivery in Mobile Networks," in *ICNP*, 2014.
[33] S. Luo *et al.*, "Ip/ndn: A multi-level translation and migration mechanism," in *NOMS*, 2018.
[34] I. Moiseenko *et al.*, "Communication Patterns for Web Interaction in Named Data Networking," in *ICN*, 2014.
[35] J. Crowcroft *et al.*, "Plutarch: an argument for network pluralism," *ACM SIGCOMM CCR*, vol. 33, no. 4, 2003.
[36] M. Jahanian *et al.*, "Graph-based namespaces and load sharing for efficient information dissemination in disasters," in *ICNP*, 2019.
[37] J. Chen *et al.*, "COPSS: An Efficient Content Oriented Pub/Sub System," in *ANCS*, 2011.
[38] M. Jahanian *et al.*, "Interoperability of ICNs and IP," 2020, https://www.cs.ucr.edu/~mjaha001/ICI-TR.pdf.
[39] V. S. Miller, "Use of elliptic curves in cryptography," in *CRYPTO*, 1985.
[40] R. van Rijswijk-Deij *et al.*, "On the adoption of the elliptic curve digital signature algorithm (ecdsa) in dnssec," in *CNSM*, 2016.
[41] P. Resnick and R. Sami, "Sybilproof transitive trust protocols," in *EC*, 2009.
[42] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: a classification," in *ISSPIT*, 2003.
[43] P. Gasti *et al.*, "DoS and DDoS in named data networking," in *ICCCN*, 2013.
[44] A. Compagno *et al.*, "Poseidon: Mitigating interest flooding ddos attacks in named data networking," in *LCS*, 2013.
[45] "MF Software Release," http://mobilityfirst.orbit-lab.org/wiki/Proto.
[46] M. Jahanian *et al.*, "Formal verification of interoperability between future network architectures using alloy," in *ABZ*, 2020.
[47] "ORBIT," http://www.orbit-lab.org/.
[48] S. Li *et al.*, "Mf-iot: A mobilityfirst-based internet of things architecture with global reach-ability and communication diversity," in *IoTDI*, 2016.