An Application-Layer Gateway for TCP/IP and Content Centric Network Interoperability

Christopher A. Wood*
University of California Irvine
Irvine, CA, USA
woodc1@uci.edu

ABSTRACT

With the growing presence of data streaming services and applications in today's Internet, content-centric networks (CCNs) are an increasingly attractive design alternative to the traditional IP-based host-oriented architecture. CCNs emphasize content by making it directly addressable and routable within the network, in contrast to addressable hosts and interfaces. This fundamental difference leads to vastly different mechanisms to publish and retrieve content and enable peer-to-peer communication. Named Data Networking (NDN) and its sibling implementation - CCNx - is one particular CCN design that has received considerable academic and industry attention. Despite the many promising benefits, there has been little research into the NDN deployment strategy. Clearly, an incremental deployment strategy is the only viable solution. During this integration phase, however, there will undoubtedly be a need for IP-based (NDN-based) hosts to communicate with and retrieve content from NDN-based (IP-based) hosts. To this end, we present an IP/NDN gateway to support interoperability between these different networking architectures with minimal application and transport layer modifications via semantic translation between the communication mechanisms used in both networks. The performance overhead induced by this gateway is studied in unidirectional and bidirectional communication settings.

Categories and Subject Descriptors

H.4 [**TODO**]: TODO; D.2.8 [**TODO**]: TODO—*TODO*

General Terms

NDN/CCN; CCN Deployment; Network Gateway

1. INTRODUCTION

At its core, the design and architecture of today's Internet is communication-based packet switching network. Since its inception, it has been retrofitted with a variety of transport and application layer protocols and middleware to support a growing set of consumer applications, such as the Web, email, and perhaps most importantly in recent times, media streaming services. The latter type of applications are bandwidth-intensive content distribution applications which leverage the underlying communication-based network as a distribution network, leading to a massive consumption of vital networking resources.

Information-centric networks (ICNs) are a new class of network architecture designs that aim to address this increasingly popular type of network traffic by decoupling data from its source and shifting the emphasis of addressable content from hosts and interfaces to content [1]. By directly addressing content instead of hosts, content dissemination and security can be "distributed" throughout the network in the sense that consumers requests for content are satisifed by *any* resource in the network (i.e., not necessarily the original producer). For example, routers close to consumers may cache content with a particular name and then satisfy all content requests that match the content's name. In-network caching and data-centric security measures are two of the defining characteristics of these new network designs.

As of today there are several information-centric networking proposals being explored as alternative designs to today's Internet; Named Data Networking (NDN) [2] is one of the more promising designs that is still an active area of active research (see `www.named-data.net` for more information). As a replacement for IP-based networks, the complete adoption of NDN, or any one of these designs, will realistically need to be done by slow and continual integration and replacement of IP-based networking resources with NDN-based resources. Currently, however, there is no engineering plan to support the IP-to-ICN integration without significant software modification and application, transport, or network layer source code modifications.

Consequently, the primary objective of this work is to aid the integration of future content-centric networking resources into the existing IP-centric Internet by providing an application and transport layer gateway between IP and NDN resources. We assume that NDN

*NSF GRFP blurb.

will not be deployed over the IP network, as such a deployment scheme would nullify the need for a gateway between the two networks. Application-layer traffic corresponding to protocols such as HTTP, FTP, SMTP, IMAP, etc. will be translated by middleware running in such gateways to correctly interface with the NDN resources, thereby serving as a semantic bridge between these two fundamentally different networking architectures. Similarly, using a custom NDN-to-IP protocol, NDN *interests* for content will be translated to the messages adhereing to the corresponding application-layer protocol to traverse the IP network. When serving as a bridge between islands of NDN resources, two gateways will leverage the features of the IP network to move messages among two separate NDN beds. Specifically, NDN interests may be encapsulated in UDP datagrams composed of IP packets traversing an IP network between two NDN islands. The primary benefit of this semi-transparent gateway is that existing IP-centric applications need not be modified at any layer in the network stack to interoperate with NDN resources.

Emphasize notlike a broker

2. NDN/CCNX OVERVIEW

NDN is one of the three currently active NSF FIA (Future Internet Architecture) projects. CCNx is a closely related and more commercially-oriented design backed by PARC. The defining characteristic of both designs is that they decouple location of content from its original publisher. To obtain content, a consumer issues a request (called an *interest*) referencing the name of the desired content. An interest is routed, based on the specified name, rather than a destination address, over a sequence of routers, each of which keeps state of the forwarded interest; see below. Requested content might be found either in a router that cached it based on a prior interest, or at the producer. Regardless, each router is expected (though not mandated) to cache each content it forwards. In essence, router caches and addressable content enable NDN/CCNx to reduce congestion and latency by keeping content closer to consumers.

An *interest*, though intended to carry a meaningful (human-readable) URI-like name, can in fact carry arbitrary strings corresponding to any type of data, such as encoded binary. Upon receiving an interest, a router looks up the content by name in its *content store* (CS), i.e., a cache. A match in the CS causes the associated content to be forwarded downstream over the same interface upon which the interest arrived and no state is kept. Interests that do not match any cached content are stored in a *Pending Interest Table* (PIT) together with the incoming and the outgoing interfaces. The interest is forwarded based on the longest-prefix match in the local *Forwarding Information Base* (FIB) table. Multiple interests matching the same name are collapsed into a single PIT entry to prevent redundant interests being sent upstream. Once a content matching a PIT entry is received by a router, it is cached (unless the interest explicitly asks not to cache this content) and forwarded to all incoming interfaces associated with the PIT entry. Finally, the PIT entry is flushed.

Beyond the pull-model that guarantees symmetric interest and content flow, content-centric traffic in NDN and CCNx has strong security implications. **First**, security is tied to content, as opposed to the channel through which it flows. All sensitive content must therefore be encrypted to ensure confidentiality. Content integrity and origin authentication are guaranteed by mandatory digital signatures; all content producers are required to sign content before responding to interests. Due to performance considerations, content signature verification, though possible, is not required by routers. Whereas, consumers are expected to verify all content signatures. Issues regarding signature verification and public key as well as trust management are discussed at length in [**?**]. **Second**, a lack of source and destination addresses in interest and content packets benefits privacy and anonymity. However, as discussed in the following section, this is insufficient to attain strong anonymity.

3. MOTVATING INTEROPERABLE HETEROGENEOUS NETWORKS

Consider the typical hourglass network stack in IP-based networks as shown in left-hand image of Figure 3. This layered design with a thin-waist infrastructure (IP packets for traffic flow) is what enabled the Internet to grow and expand at such a rapid rate; higher layers in the protocol stack extend this communication medium with support for a variety of applications and networking features (e.g., reliable message traversal via TCP). While the NDN architecture introduces a fundamental paradigm shift in the way information is published and retrieved on a network, its design, shown at a high level in the right-hand image of Figure 3, borrows the same hourglass design as IP networks. Observe that upper layers of the network stack still promote the development of robust applications based on the underlying communication layers. The difference, however, is that network traffic flow management (i.e., to enable reliable and stable communication) and security are *built into* the network stack. These architectural differences mean that application, transport, and network layer protocol semantics in IP-based networks are distinct from protocol semantics in NDN networks. The NDN gateway is intended to bridge between IP and NDN networks by performing this semantic translation between protocols.

Assuming that NDN is not to be deployed over IP,

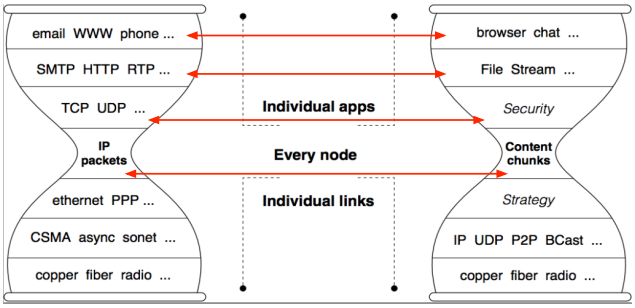

Figure 1: A visual comparison of the network stacks for the IP and NDN network architectures.

but instead as a separate network stack entirely, the need for such a gateway cannot be understated. Consider two instances of an application, A and B, that wish to send data back and forth to each other. Application A is running on a host with only an IP interface, and application B is running on a host with only an NDN interface. What does it mean for application A to establish a TCP connection stream with application B and what does it mean for application B to issue an interest to application A when neither application speaks the network language of the other. In order for these two applications to communicate, the semantics of a TCP stream-oriented connection must be translated to a stream of contiguous interests, and vice versa. Now consider the alternative scenario in which two or more physically disjoint NDN networks need to communicate to distribute content, but such networks can only be connected via the Internet. Strategically placed border gateways at the edges of these NDN networks can be used to establish a bridge across the Internet to enable cross-network interest issuance and content retrieval. Given these two seemingly unavoidable scenarios in an incremental deployment of NDN networks, our IP/NDN bridge will enable continual application operation with minimal application and transport layer software changes. In the following sections we describe the gateway design that faciliates such interoperability.

gateway is two-sided facade

4. NETWORK SEMANTIC TRANSLATIONS AT THE GATEWAY

Traditional IP-based applications and upcoming NDN-based applications treat both the network and content in significantly different ways. In IP-based settings, application and transport layer mechanisms and protocols leverage the underlying IP network layer to send packets to specific hosts. In contrast, in NDN-based settings, there are no straightforward application or transport layer analogs; the network layer, responsible for the issuance of interests and retrieval of content, is abstracted to authenticated (i.e., digitally signed) content objects or streams of data that are consumed by applications. From this perspective, there is no clear bijection between application and transport layer communication in IP-based settings and content and stream-centric data retrieval in NDN-settings. Therefore, to support the interoperability of these two networking paradigms, we need a mechanism to translate the semantics of IP-based communication to and from NDN-based content retrieval.

The direction of traffic across the gateway has a strong influence on how this semantic translation is done: IP-to-NDN application and transport layer protocols will be mapped to corresponding NDN-interests, and NDN-to-IP traffic (in the form of interests) will be encoded so as to map to the appropriate IP-based application or transport layer protocol. Stateless protocols such as HTTP greatly simplify the job of the gateway because it need not maintain any state to support communication across both networks. However, stateful protocols, such as TCP, naturally require the gateway to maintain state so as to emulate the behavior of an endpoint or host that implements such protocols. For instance, if an IP-based application wishes to establish a TCP connection with an application running on an NDN network to retrieve data, then the gateway must maintain stateful information needed to transform streams of data retrieved over a TCP socket to (packed) discrete and contiguous interests in the NDN network. In what follows we describe the semantic translation details for application and transport layer stateful and stateless protocols in both directions across the gateway.

4.1 IP-to-NDN Semantic Translation

TODO: could link in CCNx library and re-write networking code, but that requires too much work, instead we re-use existing protocols for encoding interests

4.1.1 Application Layer

Translating IP-based application layer messages to NDN interests is highly dependent on the particular application protocol in question. There is an intuitive NDN-friendly encoding of HTTP GET requests in which human-readable URIs are parsed as interest names. Stateless application-layer protocols enable such direct mappings. Therefore, the gateway supports IP-to-NDN application-layer messaging by encoding interests in HTTP GET requests as follows. NDN content objects with names "ccnx:/name/of/content" are issued via the gateway by sending a HTTP GET request with the following format to the gateway (in this example, the IP address X.X.X.X of the gateway is known or can be obtained

via DNS):

```
GET X.X.X.X:80/ndn/ccnx/name/of/content
```

Since HTTP requests and NDN interests are stateless, the gateway will parse the URI of the request to determine that (1) it corresponds to an NDN interest, and (2) the full interest name is explicitly "ccnx:/name/of/content". Upon reception, the gateway will store the key-value pair (name, (sourceIP, sourceport)) in the IP pending message table and issue an interest with the given name to the NDN network. Upon receipt of a piece of content, a NDN content handler callback recovers the IP address and port from the pending message table using the content name as the index and then writes the raw content back to the client over the same incoming HTTP TCP connection. Since it is not required that the HTTP request uses a persistent TCP connection, the HTTP message handler is a synchronous procedure that blocks while the NDN interest is satisfied so that the same TCP connection can be used to write the response.

4.1.2 *Transport Layer*

Since interests can be overloaded to contain arbitrary data, our gateway design exploits this characteristic to carry transport-layer streams of data from applications on IP hosts to supporting applications on NDN-hosts. IP is a host-based protocol, however, and so establishing a "virtual" TCP connection between two such applications must be done in two steps: (1) a TCP socket from the IP client to the gateway must be established, and (2) the first chunk of data sent from the client must be the NDN host prefix to which data will be sent. The gateway parses this path and stores it internally for each open TCP socket. Let C be the IP client generating data in the socket S, G be the gateway forwarding data on behalf of C, and prefix be the NDN host prefix sent during the connection establishment phase. All subsequent data chunks are then forwarded to this NDN host using the following steps:

1. C sends a chunk of B bytes to G over the socket S.
2. G reads B bytes of data from S and encodes it in Base64 format, denoted as data.
3. G generates a uniformly random string nonce of k bits from $\{0,1\}^k$.
4. G builds the interest prefix/nonce/data and issues the interest to the NDN network.

By incorporating a fresh random string in each new interest name, the probability that any issued interest will be satisfied from a network cache instead of the desired NDN host is negligible. Therefore, all data sent over the socket S will reach the NDN host, and the receiving application can parse the last component of the interest as fresh data. Note that since this is strictly IP-to-NDN communication, there is no state information persisted in the session table. This is because the NDN host application may not respond with data since the gateway is forwarding *transport* layer data, unlike the case where application-layer queries are forwarded.

4.2 NDN-to-IP Layer Semantic Translation

Due to their architectural differences, there does not exist a native correspondence between NDN interests and IP-based application layer protocol messages. For example, there is no standard way for a client to represent an HTTP GET request in the format of an NDN interest. To make this type of semantic translation possible, the NDN-to-IP application layer bridge leverages the human-readable names of content to encode IP-based application layer protocol specifics. The encoding for HTTP and FTP semantic translations is specified in EBNF form below; other application-layer protocols can easily be supported by extending this grammar in the natural way.

Command pattern

Interests encoded using this grammar are intercepted in the NDNInputStage of the gateway (see Figure 6). Upon reception, the gateway parses the message, stores a new entry in the NDN pending message table, and forwards the decoded message contents to the appropriate IPOutputStage pipeline stage. Upon reception, the IP response is retrieved in the IPInputStage and forwarded inwards to the NDNOutputStage, where the corresponding entry in the NDN pending message table is indexed using the contents of the arriving message to retrieve the original incoming interest name. Once fetched, the gateway creates and signs a new content object with the IP network response, and then forwards the content downstream to its intended consumer.

Unlike traditional NDN routing, the gateway pending message table does not collapse interests by default. The reason for this is that application-protocol interests are often issued when *new* state needs to change (i.e., cached responses not generated on demand from the intended IP recipient are not acceptable).

It is important to note that stateful protocols such as TCP must explicitly embed identifying information about the consumer in order to operate correctly. The reason is that one TCP stream must be associated with at most one consumer, and since NDN does not have any notion of host addresses or identifiers, the consumer application must explicitly encode its identity in the interest. Our grammar enforces identities based on consumer public keys and a corresponding digital signature of the entire interest for such protocols so that consumers can be explicitly identified and their sessions cannot be hijacked by other consumers (doing so would require compromising a consumer and its private key).

⟨*ip-interest*⟩→ '/. . . /ip/'⟨*protocol*⟩

⟨*protocol*⟩→ 'http/'⟨*http-cmd*⟩['/'⟨*http-path*⟩] | 'tcp/'⟨*uri-encoded-string*⟩'/'⟨*tcp-ident*⟩

⟨*http-cmd*⟩→ 'GET' | 'PUT' | 'POST' | 'DELETE'

⟨*http-path*⟩→ ⟨*uri*⟩ | ⟨*ip-address*⟩[port]['/'⟨*uri-encoded-string*⟩]

⟨*tcp-ident*⟩→ ⟨*SHA256-hash*⟩'/'⟨*public-key*⟩

⟨*tcp-param*⟩→ ⟨*uri-encoded-string*⟩

⟨*uri-encoded-string*⟩→ TODO

5. BRIDGING ISOLATED NETWORKS VIA MESSAGE ENCAPSULATION

The second type of interoperability scenario that may arise during the deployment of NDN networks is when two or more physically disjoint NDN networks need to communicate with eachother. Let I_i and I_j be two such dijoint NDN networks, and let A_i and A_j be two applications running on hosts in I_i and I_j, respectively. Without any additional mechanisms, A_i and A_j would not be able to communicate. However, if there existed two gateways at the edges of I_i and I_j, each of which connected to the same IP-based network (i.e., the Internet), then interests from A_i (A_j) could be sent to A_j (A_i) as follows:

1. An interest from A_i is intercepted a gateway G_i.
2. G_i encapsulates the interest in an IP packet sent to gateway G_j adhering to IPsec.
3. G_j unwraps and re-issues the interest and waits for the content to be retrieved from A_j. Upon reception, the content's signature is verified and the content is encapsulated in a signed IPsec packet and sent to G_i.
4. G_i verifies the signature of the packet, creates and signs a new content object, and sends the content object back downstream to A_i.

A visual depiction of this scenario is shown in Figure 5. Of course, many optimizations can be made to this simplistic design to increase interest throughput across the bridge. First, persistent TCP connections are maintained between adjacent gateways.

-
- Interests are routed based on the physical distance supporting gateways.

TODO: handshake between gateways to establish secure tunnel (SSL/TLS), protocol that lists all explicit steps,

Algorithm 1 EstablishBridge

Require: Anonymous routers $r_1, r_2, \ldots, r_n$ $(n \geq 1)$ with public keys $pk_1, pk_2, \ldots, pk_n$.

6. IMPLEMENTATION OVERVIEW

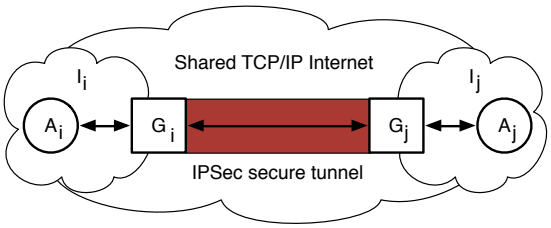

Figure 2: TODO.

pipeline stage defers parsing to subclasses - template pattern

7. PERFORMANCE EVALUATION

8. REFERENCES

[1] 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*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[2] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

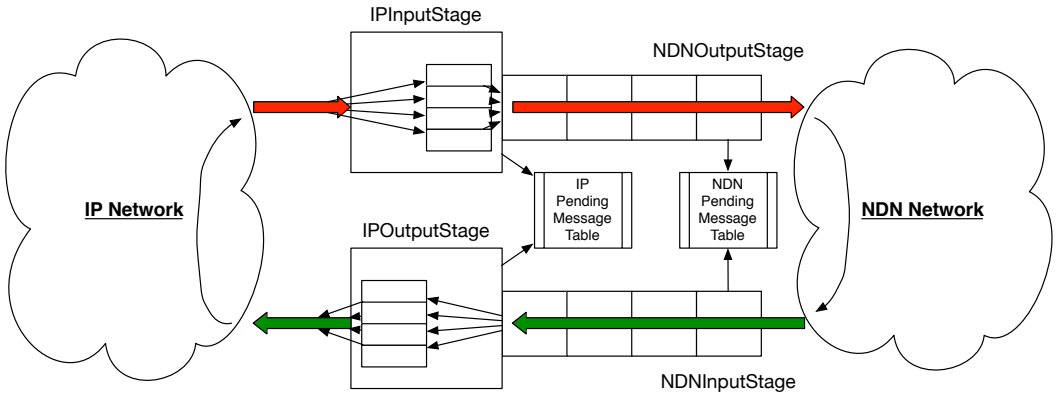

Figure 3: Bidirectional message pipeline for IP-to-NDN and NDN-to-IP message traversal.