Named-Data Transport: An End-to-End Approach for an Information-Centric IP Internet

Abdulazaz Albalawi
Computer Science and Engineering Department
University of California Santa Cruz
Santa Cruz, CA, USA
aalbalaw@ucsc.edu

J.J. Garcia-Luna-Aceves
Computer Science and Engineering Department
University of California Santa Cruz
Santa Cruz, CA, USA
jj@soe.ucsc.edu

ABSTRACT

Named-Data Transport (NDT) is introduced to provide efficient content delivery by name over the existing IP Internet. NDT consists of the integration of three end-to-end architectural components: The first connection-free reliable transport protocol, the Named-Data Transport Protocol (NDTP); minor extensions to the Domain Name System (DNS) to include records containing manifests describing content; and transparent caches that track pending requests for content. NDT uses receiver-driven requests (Interests) to request content and NDT proxies that provide transparent caching of content while enforcing privacy. The performance of NDT, the Transmission Control Protocol (TCP), and Named-Data Networking (NDN) is compared using off-the-shelf implementations in the ns-3 simulator. The results demonstrate that NDT outperforms TCP and is as efficient as NDN, but without making any changes to the existing Internet routing infrastructure.

CCS CONCEPTS

• **Networks** → **Naming and addressing**; **Network protocol design**; **Transport protocols**.

KEYWORDS

ICN, NDN, DNS, Transport protocols, TCP

1 INTRODUCTION

The limited support for reliable and efficient access to (e.g., videos and documents) and services (e.g., multiplayer games) provided by the communication protocols of the IP Internet led to the development of dedicated systems aimed at content delivery. These systems can be categorized as content delivery networks (CDN) [12, 104] like Akamai [79] or peer-to-peer (P2P) applications [67] like BitTorrent [14]. While such content-delivery systems address many of the performance limitations of the basic Internet protocol stack, they require either third parties to provide added functionality or applications to replicate content-delivery functionality independently of others. As a result, several Information-Centric Networking (ICN) architectures have been developed to address the limitations of the IP Internet and the inefficiencies of using CDN's and P2P applications. These architectures [2, 11, 103] aim to allow all Internet applications and users to obtain content and services by name, without requiring specific systems to serve different applications or forcing applications to determine where instances of content and services of interest are located in the Internet.

Section 2 provides a summary of information-centric approaches proposed in the past and highlights four important facts that motivate the subject of this paper. First, prior transport protocols leave applications in charge of reliable-communication functionality, rely on connections to provide end-to-end reliable communication, or require a network layer that provides much more than datagram delivery. Second, CDN and P2P approaches are insufficient to address the ever-increasing content-oriented usage of the Internet by *all* its users and applications. Third, most ICN architectures attain efficient content distribution by: naming content and services rather than endpoints; using in-network caches to allow content to be delivered opportunistically from closer locations; and using either name-based routing and embedding name resolution in the routing infrastructure, or overlay networks of name resolvers or content routers. Fourth, prior approaches that focus on modifications of the Domain Name System (DNS) to accommodate content distribution assume the use of TCP connections.

Name-based routing and forwarding have been considered to be necessary to attain an information-centric Internet. The most popular example of this approach today is the Named-Data Networking (NDN) architecture [76]. Needless to say, NDN and other ICN architectures have produced valuable insight on what it means to provide information centricity at Internet scale. However, by requiring a far more complex network layer, these approaches have deviated from the end-to-end principle [10, 87] and the subsequent end-to-end arguments [15] that made systems built on datagram packet-switching networks so successful, including the Internet. Using name-based routing to attain an information-centric Internet is an architectural choice–not a requirement–that has profound implications. It means that efficient content delivery *must* come at the cost of redesigning the IP routing infrastructure and doing away with the DNS. Furthermore, servers and clients must also be modified to adopt the API's needed to use a name-based network layer, and widely successful application protocols like HTTP must be redone. The contribution of this paper is to introduce an alternative end-to-end approach to an information-centric Internet that we call **Named-Data Transport** (**NDT**).

Conceptually, NDT amounts to an end-to-end implementation of the Interest-based approach first introduced by Jacobson et al. [60]

ICN '20, September 29-October 1, 2020, Virtual Event, Canada

ACM ISBN 978-1-4503-8040-9/20/09…$15.00
https://doi.org/10.1145/3405656.3418714

that does not require any changes to the Internet routing infrastructure and takes advantage of the DNS. NDT allows applications to ask for content and services by name, and makes the transport layer responsible for: (a) mapping content names to one or multiple locations where the content is offered, (b) delivering the content reliably to consumers from content servers or transparent caches without using end-to-end connections, and (c) enforcing the privacy of consumers. Like other ICN architectures, NDT can provide the benefits attained with CDN's and P2P applications, but without requiring the services of specific vendors or replicating functionalities at the application layer. Furthermore, NDT can be deployed incrementally and much faster than prior ICN architectures.

Section 3 presents the **Named-Data Transport Protocol**, or **NDTP**, which is the first transport protocol that provides reliable end-to-end communication over a datagram communication infrastructure without establishing connections. NDTP does this by using content names and manifests that describe the structure of the content as in some ICN-based architectures [13, 74, 100]. In NDT, applications have access to content names in the same way clients have access to Uniform Resource Locators (URLs) today. Applications request content by name and NDTP provides reliable name-based transport services by obtaining the structure of content objects from the DNS without the need for end-to-end connections and in a way that is transparent to end-user applications.

Section 4 describes how content names are mapped into manifests by the **manifest-yielding DNS** (**my-DNS**), which augments the DNS to maintain **manifest records** describing the locations and structures of content objects. Given that NDTP consumers obtain the location of the nearest copy of a content object given its name, my-DNS serves as a de facto name-based routing overlay with redirection operating on a publish-subscribe basis. Using my-DNS redirection eliminates the need for name-based routing protocols while incurring very small additional delays.

Section 5 describes how NDT uses **NDT proxies** to: (a) support transparent caching of content with privacy over the IP Internet for arbitrary applications, (b) secure content relying on informational asymmetry to prevent caches from accessing cached content and on computational asymmetry to prevent clients from decoding unauthorized content, and (c) support multicasting services without the need for IP multicast routing. NDT proxies maintain Pending-Interest Tables (PIT) similar to those used in NDN and other ICN architectures, but without making any changes to the IP Internet routing infrastructure.

Section 6 compares the performance of NDT with the performance of TCP and NDN using our implementation of NDT in ns3, which is publicly available [4] to foment further research on NDT, and off-the-shelf implementations in ns-3 of TCP and NDN. The results of the simulation experiments show that NDT is inherently more efficient than TCP and as efficient as NDN, but without the need for a clean-slate redesign of the Internet routing infrastructure. Section 7 provides our conclusions.

2 RELATED WORK

We summarize prior work related to NDT, including transport protocols, CDN and P2P approaches, ICN architectures, support for mobility and privacy, and end-to-end DNS-based schemes.

2.1 Reliable Transport Protocols

TCP and all subsequent transport protocols that provide reliable end-to-end communication are based on end-to-end connections [54, 64, 80, 83]. This has become a big problem for content-oriented Internet applications because connections are brittle, in most protocols the context exchange must be restarted if a connection is lost, a specific site must be selected to start the connection supporting content retrieval, and in-network caching cannot be used without compromising privacy.

QUIC [64] uses connection identifiers for additional resilience to physical connectivity losses and mobility. However, it has been shown that no correct connection-based protocol exists that provides reliable communication in the presence of nodes failing and losing their state [69, 70, 94]. Hence, guaranteeing the correct operation of *any* connection-oriented protocol requires that either the sender and receiver participating in a connection have non-volatile memory to remember any identifier used previously, or that connection identifiers are never reused to identify incorrectly past connections.

2.2 CDN and P2P Approaches

The limitations of using end-to-end connections between specific sites to support content delivery prompted the development of many CDN and P2P approaches over the years. Many surveys and taxonomies have been published for CDN's and P2P applications and overlays (e.g., [12, 67, 68, 104]), and it should be clear that such systems provide much if not all of the functionalities and services present in *all* ICN architectures. For example, name-based content routing and redirection strategies were implemented in CDN's (e.g., [44, 52, 68, 85]) before they were advocated in ICN architectures.

The key difference between ICN architectures and CDN's or P2P applications is that they are applicable to *all* Internet users and applications, rather than just those users serviced by specific CDN third parties or specific P2P applications. It should be noted that the end-to-end nature of NDT enables the use of many CDN and P2P techniques to further improve its efficiency without requiring changes to the Internet routing infrastructure.

2.3 Prior ICN Architectures

TRIAD [53] and Content-Based Networking [20–22] were the earliest examples of ICN architectures advocating content-based routing by names or attributes and publish-subscribe operation. Many subsequent ICN architectures evolved over the years that either proposed to change the network layer of the Internet to implement location-independent name-based routing and forwarding, or adopted the use of overlays as in CDN's and P2P systems to provide name-based routing, resolve names to locations of content, and support publish-subscribe functionality. These ICN architectures include: COMET [36, 37], Content-Centric Networking (CCN) [60], CONVERGENCE [42], DONA [63], PSIRP and PURSUIT [38, 39, 102], MobilityFirst [78, 89], Named-Data Networking (NDN) [76], Nebula [6], 4WARD [41], SAIL [40, 96], and XIA [56].

The advantages and disadvantages of the ICN architectures are discussed in several surveys (e.g., [1, 2, 5, 11, 103]). Over the years, NDN has become the most successful ICN architecture to date, and more recent work has even focused on embedding NDN routers as

part of the IP Internet routing infrastructure [19, 75]. Accordingly, we use NDN as the leading example of prior ICN architectures.

The main benefits of NDN over the IP Internet architecture that have been stated in the past are: (a) Allowing all Internet applications to request content and services by name rather than locations, (b) eliminating performance issues associated with TCP connections, (c) providing added privacy to content consumers, (d) making efficient use of in-network caching for arbitrary content, and (e) enabling multicast services without the need for multicast routing protocols. The subsequent description of NDT and its comparison with NDN show that NDT provides all these benefits without the need to change the Internet routing infrastructure or establish new overlays on top of it.

In contrast to NDN and other ICN architectures, NDT does not require the deployment of overlays of name resolvers or content routers, or changes to the Internet routing infrastructure to operate. This approach is motivated by recent results on caching and ICN schemes. There is ample evidence that edge caching provides most of the benefits derived from in-network caching at every router [27, 28, 35]. On the other hand, the number of conent name prefixes needed to provide name-based routing at Internet scale is many orders of magnitude larger than the number of Internet IP address ranges needed to support address-based routing. Hence, independently of the type of name-based content routing used [45, 51, 57, 99], the combined use of address-based routing and redirection schemes that map names to addresses can be done far more efficiently than the combined use of name-based routing and PIT's at each router. Performance results of recent ICN approaches based on addresses indicate that this is the case [46–50].

2.4 Privacy and Mobility Support

Prior approaches attempting to address the privacy concerns of transparent caching using end-to-end connections are not effective. Traffic carried over closed secure connections used in HTTPS cannot be cached, and this is a growing concern for content retrieval on the Internet because more than 50% of web traffic is served over HTTPS [31]. GroupSec [92] adapts HTTPS to support group memberships to allow transparent caching on the Internet without making any changes to caches or servers. Unfortunately, this approach does not provide privacy among members in the same group, and an adversary can infer whether multiple clients are in the same group. Prior ICN architectures address privacy with different mechanisms [5], and Arianfar et al. [7] propose a privacy technique for NDN that leverages computational asymmetry to prevent caches from decoding cached contents in real-time; however, this method does not provide complete privacy for consumers.

The use of end-to-end connections between specific end-point addresses prompted the development of many approaches to cope with mobility in the context of the IP Internet [9, 33, 43, 65, 82, 86, 91, 93]. However, this prior work does not address eliminating connections for reliable end-to-end communication, and some can be used in NDT.

2.5 DNS-based Approaches

iDNS [90] and idICN [35] are arguably the first proposed approaches aiming to provide the ICN benefits in the current Internet by leveraging the DNS system. Instead of resolving a URL hostname to an IP address, iDNS resolves content names directly to metadata that contains the address of servers hosting the content, taking into account local caches; however, iDNS does not specify a specific protocol to retrieve contents using its method and instead leave it as part of their future work.

idICN also uses the DNS to resolve a content name to a nearby cache or a hosting server, and uses HTTP as the baseline for the transfer protocol. The limitations of this approach are that it is limited to HTTP, requires the connections used in TCP, and which must take place at the application level, causing different types of proxies to be used for caching. Similarly, DNS++ [30] and NEO [34] introduce an information-centric API combined with a DNS-based mechanism very similar to the one used in iDNS, and have the same limitation of requiring TCP and legacy application protocols to operate.

3 NDTP

NDTP eliminates the need to maintain the ephemeral type of context provided using connections by allowing a consumer and producer to share a common description of the structure of the content being exchanged, and such that both can refer to that description to deliver specific portions of content reliably. This description is called the **manifest** of the content object being delivered. The manifest of a content object frees the NDTP consumer and producer of the object from having to create and maintain the context for their reliable exchange of the object in real-time.

Provided that an NDTP consumer and a producer can refer to the same manifest, they can exchange any portion of the content object described in the manifest on a transnational basis, and an NDTP consumer process is also free to contact multiple parties hosting the content using the same manifest published by the NDTP producer of that content. More specifically, the NDTP process managing application content publishes manifests accessible on the IP Internet that describe how the content can be retrieved. The server application servicing the content notifies its NDTP producer to publish a manifest that maps to a unique global name and describes how the content object is structured. Before the NDTP process servicing the client application starts retrieving the content object, it queries its local my-DNS (manifest-yielding DNS) to resolve the object name to its manifest record, which contains the manifest of a content object and additional information to manage the content on the IP Internet. Once the consumer obtains the manifest record for the content object, it proceeds with a window-based sequence of Interests requesting the chunks that are needed to decode the content object, as stated in the manifest record.

For convenience, NDTP is implemented over UDP, just like several recent transport protocols. As a result, NDTP operates completely in user-space, and its packets are encapsulated in UDP datagrams, which are encapsulated further in IP datagrams. NDTP is implemented using two main data structures: A producer that publishes content and a consumer that retrieves content. This design is inspired by the NDN API design [72].

3.1 Publishing Content

Figure 1 shows an overview of the NDTP producer functional structure. When a server application needs to publish content, it is the responsibility of the NDTP producer to publish the content on the

Internet. This consists of three main parts: (1) saving the content object and its name into its content store, (2) sending requested data packets in response to received Interests, and (3) publishing a manifest record for the content object by registering it along with its name with its authoritative my-DNS server.

Before publishing content on the Internet, the producer segments the content into multiple chunks and encodes them in a specific way to ensure the privacy of cached content. An example of an encoding method is explained in Section 5.1. Chunks can also be signed and encrypted to ensure security at this stage. Once the content is segmented into multiple chunks, it is cached at the content store. The content store can be viewed as the sender buffer in TCP and other connection-based transport protocols. The producer then appends the names of the chunks into the manifest along with the encoding method and other security parameters. The final stage of publishing content on the Internet is by publishing its manifest. This is done by constructing the manifest record using the manifest itself and meta-information about the content (e.g., a time to live, list of servers to contact, etc.). The producer then registers the manifest record along with the content name with a my-DNS authoritative server. Registering and updating manifest records is done using regular DNS standards [98], and it is up to the content provider to determine which sites to use to host content objects.

An Interest from a consumer goes through the Interest processing routine, which is used to perform a cache lookup at the content store. The Interest is simply dropped if it's a miss, and a data packet is sent back if it succeeds. A NACK can be sent back to the consumer if necessary. The design of NDTP allows application developers to customize their content distribution and have full control over deployment decisions.

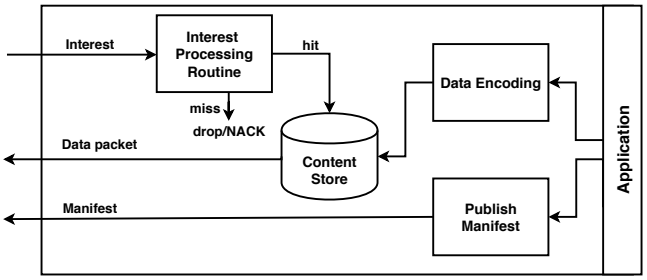


Figure 1: Processing Interests and data packets at an NDTP producer

3.2 Retrieving Content

Figure 2 shows the functional structure of an NDTP consumer. Client applications running over NDTP retrieve data objects using their names (URL). The client only needs to provide the content-object name to the NDTP layer. The NDTP consumer does all the work in retrieving the content, which involves contacting the local my-DNS to retrieve the manifest record, and requesting the actual content and decoding it. Using the NDTP API, clients access the content directly through the function `GetContentByName()`, which takes the content name as its parameter. Calling this function invokes the consumer side of the NDTP layer.

The NDTP consumer responsibility can be broken into two main tasks: resolving content names to their manifest record, and retrieving the content using information from the manifest record. As Figure 2 shows, the consumer in NDTP remembers a set of variables for each content that needs to be retrieved. These variables are stored in a data structure called the Content Control Block or CCB, which is used to control such things as Interest timeouts, window size, and the decoding method. All Interests go through the Interest crypto routine. The routine signs these Interests based on information from the manifest. Arriving data packets will go through the Data verification routine for authentication including manifest as well as checking if packets are corrupted.

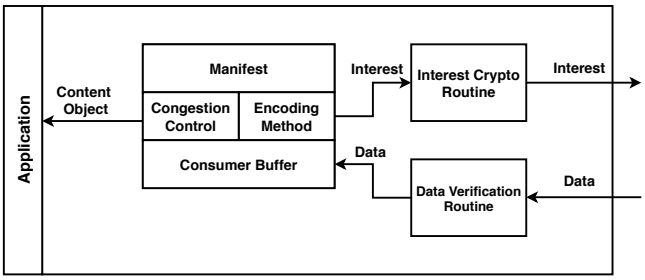


Figure 2: Processing Interests and data packets at an NDTP consumer

3.3 Retransmission and Congestion Control

Retransmission and congestion control algorithms in NDTP are receiver-driven, just as in NDN and CCN [60]. These algorithms are based on the inclusion of a manifest pointer in each Interest and data packet. Clearly, many receiver-driven strategies can be implemented. For simplicity, however, we chose to use the TCP congestion-control and retransmission algorithms in NDTP. Given the similarities of the mechanisms used in NDTP with well-known mechanisms in TCP, the rest of this section provides only an outline of the retransmission and congestion-control mechanisms in NDTP.

3.3.1 ***Congestion Control***. The NDTP consumer controls the flow of data traffic by controlling the sending rate of its Interests using a congestion window. The congestion window specifies the number of outstanding Interests allowed to be sent before receiving their data packets. The window size is adjusted based on the AIMD (Additive Increase Multiplicative Decrease) mechanism commonly used in TCP for the congestion window. The NDTP consumer increases its congestion window based on slow-start, starting with transmitting one Interest and increasing the congestion window by one for each newly received Data packet. The slow-start continues until the window reaches the slow-start threshold. The consumer operates under the congestion-avoidance state as in TCP [59] When the slow-start threshold is exceeded, and increases its window by one Interest every round-trip time.

3.3.2 ***Fast Retransmit***. NDTP uses a receiver-driven selective-repeat retransmission strategy. An NDTP consumer retransmits a lost Interest once an out-of-order data packet is received based on the order in the transmitted list if a time constraint is met. A

lost Interest y initially transmitted at time t_i is retransmitted once the following constraint is met: As soon as a data packet arrives for any Interest transmitted at t_x where (t_x>t_i), and ($t_{current}$-t_i)> RTT, where $t_{current}$ is the current time and RTT is the time it takes to send an Interest and receive the Data packet for it. Once NDTP detects a packet loss using fast retransmit, the consumer reduces its congestion window by one half and sets the slow-start threshold to the new window size causing the consumer to go into congestion-avoidance mode.

3.3.3 ***RTO Estimate.*** Because of transparent caching, congestion detection based on retransmission timeouts (RTO) is not reliable in NDN when data are retrieved from multiple sources. Many of the congestion-control protocols proposed for NDN [24] argue against the use of a single RTO to detect a packet loss because consumers cannot detect when data are being retrieved from different locations. Unlike NDN, a consumer in NDTP relies on IP addressing to identify the source of each data packet, even when data are being retrieved from multiple sources. This allows NDTP to provide accurate RTO estimates by measuring the correct round-trip time for every data packet, while many NDN congestion-control algorithms (e.g., [3, 88] must guess the sources of data packets. In the case of a timeout event, the NDTP consumer retransmits the Interest that caused the timeout, reduces its congestion window to one Interest, sets the threshold half the congestion window size before the timeout, and then goes into slow-start mode.

4 MANIFEST-YIELDING DNS

NDT attains location-independent content naming through the integration of name resolution with the transport protocol used to carry content reliably. A new resource record type, which we call *manifest record* is added to the DNS, resulting in the **manifest-yielding DNS (my-DNS)**. Instead of creating a new type of DNS resource record for the manifests, it is possible to encode the manifest using a TXT record instead. A manifest record describes the content structure by carrying the manifest generated by the NDTP producer, lists the IP addresses of the different locations of the content on the Internet, and other information, such as fields specifying the freshness of the content and fields specifying security parameters.

Content naming in NDT is inspired by the iDNS approach [90] to separate the content name from its location on the Internet. Content names in NDT are based on DNS domain names, allowing them to be persistent and unique through the hierarchical nature of my-DNS. For example, the content name contentA.ucsc.edu represents contentA hosted by the DNS domain ucsc.edu. With NDTP help, each content name on the Internet is mapped to an individual manifest generated by its producer, as explained in the previous section. Having a single authority on manifests allows consumers to authenticate the origin of content on the Internet easily. To achieve near-replica routing of content, my-DNS is used to map the name of a content object to the manifest record that describes the locations and structure of the content object to the consumers on the Internet. In turn, a manifest record maps the manifest of a content object to a list of IP addresses hosting a replica of the content. Each one of these addresses is added to the list as an individual DNS type A record. my-DNS updates this list as needed. This includes sorting the list by the nearest replica based on the consumer's geographical location issuing the DNS query for the content name. Such an approach is already being used to enhance domain-name lookup on the Internet by many vendors. Because NDT uses standard DNS procedures to resolve content names to manifest records, it can rely on standard DNS procedures to dynamically register content names with its corresponding manifest records. This is similar to a website adding DNS records to its authoritative DNS server. Content servers can dynamically register the content name with their manifest record using dynamic-update DNS mechanisms [98].

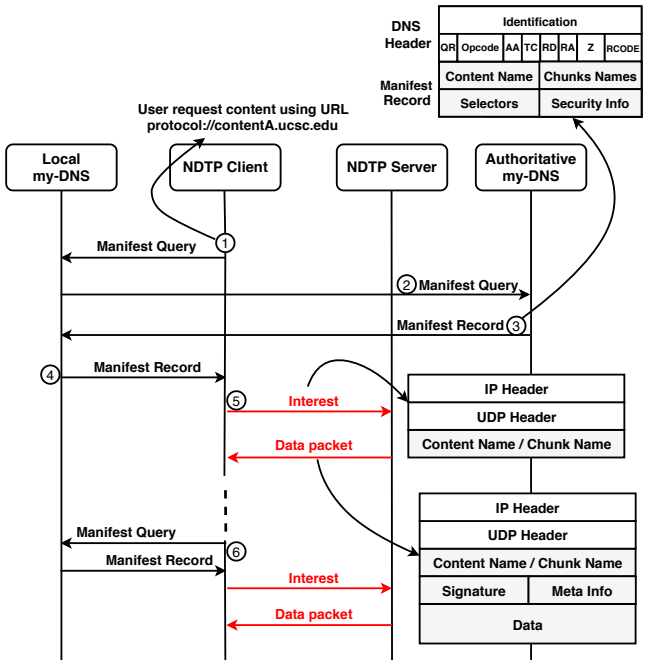


Figure 3: Mapping content names to manifest records

Figure 3 shows the steps used in NDT to resolve a content name into a manifest record using my-DNS. In Step 1, an NDTP consumer issues a manifest query with the content name passed by the client application to the local my-DNS server. The manifest query is merely a DNS query with the name field (QNAME) set as the content name and the type field (QTYP) set as the manifest record type, in addition to the standard DNS message fields. Assuming that a specific my-DNS zone manages the manifest records under its zone, the local my-DNS queries iteratively the global my-DNS for the location of the authoritative my-DNS server of the my-DNS zone specified in the URL. After the local my-DNS server obtains the IP address of the authoritative my-DNS server, it sends a query of a manifest record type along with the content name, as shown in Step 2. In response to the query from the local my-DNS, the authoritative my-DNS server returns the manifest record associated with this content name, as shown in Step 3. After receiving the manifest record from the local my-DNS server, the NDTP consumer can start issuing Interests to retrieve the content, as shown in Step 4. When another NDTP consumer tries to retrieve the same content, the local my-DNS simply returns the manifest record that has been cached, as shown in Step 5.

NDT ensures the security of the content itself, rather than relying on closed private connections. To protect the authenticity and integrity of the content objects, the manifest record must also be secured. This could be done by relying on digital signatures based on public-key cryptography as in DNSSEC [32]. However, DNSSEC is not widely deployed, is expensive to operate, and is not viewed as a complete solution [8]. New techniques are clearly needed to secure and protect manifest records, and they are the subject of future work.

Without proper care, adding manifest records to the DNS could lead to scaling problems resulting from IP address changes for content servers and mirroring sites hosting large numbers of content objects, each with a manifest record that must be stored. Fortunately, adding a layer of indirection prevents this problem, and the DNS design already provides the means to add indirection via the CNAME resource records. Using the CNAME records instead of the A records of the content server inside the manifest records, DNS updates messages to the server are avoided. Whenever a content server changes its IP address, only the A record stated in the CNAME record of the content server needs to be updated. To ensure consumers keep up with the changes of the IP address of the content server, the TTL can be set low for these records. This action does not increase the load on the authoritative DNS server, as has been discussed in the past in the context of mobile networks [101]. In addition, notification mechanisms can be used to update the local DNS with the new IP address proactively using known consistency mechanisms proposed for the DNS [25].

Mapping a domain name for every content object requires several orders of magnitude additional storage capacity in the authoritative and local DNS server. This may appear too onerous at first glance; however, as has been noted before [90], most of today's HTTP servers can handle such a load (by hosting an entire directory tree), and a dedicated DNS server can be used to host and manage manifest records for content objects under a separate domain. For example, the DNS resolution for the hierarchical content name "ContentA.Contents.example.com" would involve a maximum of four requests, with the final one to the authoritative DNS server for (Contents.example) for contentA.

We note that the possibility of incurring additional redirection delays to reduce storage requirements in my-DNS servers is a better trade off than requiring all routers to maintain FIB's and PIT's that are several orders of magnitude larger than the FIB's of IP routers today.

In terms of the size of the manifest records that are handled by the DNS, RFC 1035 [71] already defines mechanisms on how to handle large DNS responses. This is done by relying on TCP instead of UDP to handle such a response. However, another approach is to use a layer of indirection by having a manifest record pointing to other manifests that can be retrieved from the content server responsible for publishing the content objects and its manifest record instead of using the authoritative DNS server.

5 NDT PROXIES

Client requests are redirected transparently to caches in the IP Internet by intercepting TCP connections destined for specific ports or a specific set of destination addresses. This is usually done by using a layer-four switch on the route between the client and the origin server, and by splitting the TCP connection into two with a proxy web cache spoofing the connection with the client. This poses a significant privacy concern, because the cache has access to each request and response between the server and the client. Establishing private connections, such as using the TLS protocol, is not a solution because it precludes transparent caching.

NDT eliminates the limitations of transparent caching in the IP Internet by means of the manifests that describe the structure of content objects globally at the transport layer. As a result, network administrators can simply install a single NDT proxy cache in their network and configure a layer-four switch to redirect all NDTP traffic to proxies. Figure 4 shows an example of transparent caching in NDT.

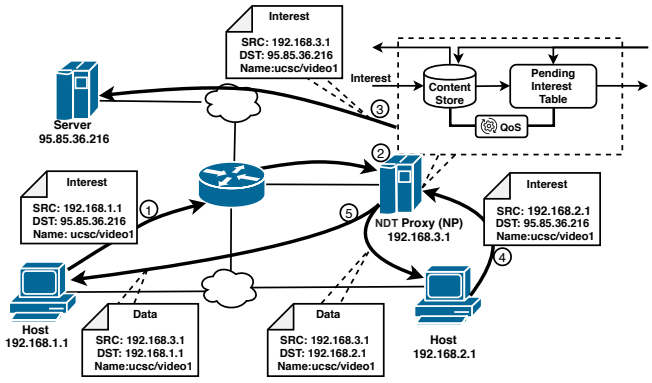


Figure 4: Transparent caching and Interest aggregation

After an NDTP consumer retrieves the manifest record of a content object, it proceeds with a window-based sequence of Interests requesting all the chunks that are needed to decode the requested content as stated in the manifest. An NDTP consumer at the client starts issuing Interests destined to the NDTP producer at the server (Step 1 in Figure 4). A layer-four switch on the way is configured to intercept Interests intended for the content server and forwards them to a nearby NDT proxy cache ((Step 2 in Figure 4). The NDT proxy then checks whether the requested data packet is stored in its content store, and forwards the Interest towards the NDTP producer using its own IP address as the origin of the Interest (Step 3 in Figure 4). Once the NDT proxy receives a data packet, it uses it to satisfy any pending Interest received from other NDTP consumers (Step 4 in Figure 4), as indicated in Step 5 of Figure 4.

NDT proxies track pending Interest from NDTP consumers using a Pending Interest Table (PIT) that serves the same purpose as in NDN. Specifically, an NDT proxy forwards an Interest only once towards the address of a content producer or mirroring site using its own IP Address as the Interest source, and aggregates subsequent Interests in the same content. However, the PIT in NDT is kept outside the routing infrastructure and does not impose additional overhead compared to today's web caches, which must keep track of TCP connections between the client and the origin server in a data structure called the connection tracker. Furthermore, forwarding Interests in NDT does not need a new routing infrastructure like NDN and similar ICN architectures do.

5.1 Securing Cached Content

From a security of standpoint, content carried over NDTP can be public or private. Protecting the privacy of public content is not a concern; however, authentication and integrity is a necessity. By ensuring the integrity and authenticity of the manifest record using methods like DNSSEC [32] as we explained previously, allows NDTP to ensure the security of the content as well. Part of a manifest record is the name of the chunks that need to be requested using Interests in order to construct the content object. By using a hashing function, NDTP uses each chunk hash digest as the name. As a result, the manifest contains the content name, the digest of each chunk composing the content, and the hashing function used by the server to name these chunks, similar to the one proposed for NDN[73]. Finally, by computing the hash digest of these chunks, an NDTP consumer can verify the authenticity and integrity of a received data packet.

NDTP relies on multiple security methods to ensure the protection of private content. The goal is to ensure privacy while also enabling transparent caching of content. To achieve this, NDTP must ensure the following: (a) Only the NDTP producer and the NDTP consumer of a content object should be able to access the content object to preserve privacy, (b) NDTP producers and NDTP consumers must be able to authenticate each other, (c) NDTP consumers must be able to detect the integrity of received content objects, and (d) NDTP should at least provide the same level of anonymity as HTTPS.

TLS and other secure methods based on end-to-end connections rely on symmetric cryptography per each connection to ensure privacy. Given that the keys they use are generated uniquely for each connection, it is apparent that using similar keys to secure content in NDTP would negate the benefits of transparent caching.

One simple way to benefit from transparent caching with symmetric cryptography is to use group keys. Such an approach certainly downgrades the privacy of those consumers retrieving the same content objects, but also ensures that caches are unable to access the content cached for the group. To increase the level of privacy between content consumers, NDTP uses computational asymmetry by encoding the data using a specific method known only to one particular consumer. This is done by combining a content object with useless data (or old chunks in the content store) intended solely for obfuscation.

5.1.1 ***Group Key.*** Using group keys should be enough to prevent an adversary from accessing cached content. However, compared to TLS in which an encryption key is used for each connection, this might raise two security issues: (1) a member inside the list may infer that other members in the same list are retrieving the same content object by observing the chunks names; and (2) members that are not part of the content list may infer that multiple consumers are requesting the same content by observing the names of requested chunk names. NDTP overcomes this problem by encoding part of the content for each consumer in a different way using computational asymmetry. Because NDTP relies mainly on old chunks to do this, the chance of these chunks being in a cache will be high, which reduces the latency to retrieve them. As a result, consumers retrieving the same content will be requesting different sets of chunks.

5.1.2 ***Encoding Data.*** The way in which NDTP encodes data to ensure computational asymmetry is by combining the content of an object with old data or "useless" data intended solely for obfuscation to produce a set of coded fragments. For example, multiple chunks of the original content can be encoded with other chunks from the content store, resulting in new coded chunks. The way in which these chunks are encoded is only shared between the content consumer and the content producer. Even though such a method does not provide complete privacy, it should make it computationally expensive for caches to access cached contents.

The method used in NDTP to encode the data of a data object is based on previous work on censorship at storage systems [95]. A similar method was also used in [7] to prevent real-time censorship at nodes in ICN networks. Algorithm 1 shows how an NDTP producer encodes content for data obfuscation.

The return manifest of Algorithm 1 is simply a list of tuples, each describing what chunks to request and how to decode the original chunk as shown in Eq. (1). After receiving its manifest, the NDTP consumer simply sends Interest for the coded chunks corresponding to the original chunk in the manifest as shown in Eq. (2).

$$\left[\left(h(B_{1,1}), ..., h(B_{c,1}), h(X_1)\right), ..., \left(h(B_{1,n}), ..., h(B_{c,n}), h(X_n)\right)\right] \quad (1)$$

$$f_i = \left(X_i \oplus_{j=1,...,c} B_{j,i}\right) \; for \; i = 1, ..., n \quad (2)$$

Algorithm 1 Data Obfuscation

```
1:  procedure ENCODE(content)
2:      f ← SEGMENT(content)                    ▷ return list of chunks
3:      for i ∈ {1, ..., n − 1} do
4:          X_i = f_i
5:          ChunkName ← NULL
6:          for j ∈ {1, ..., c} do
7:              X_i = (X_i ⊕ b_j)
8:              ChunkName ← h(b_j)
9:          ChunkName ← h(X_i)
10:         Manifest ← ChunkName
11:     return Manifest
12: ─────────────────────────────────────────
13: c: number of random old chunks
14: n: number of content's fragments
15: h: hash function
16: ChunkName: tuple of hash digest
17: Manifest: list of tuples
```

5.1.3 ***Partial Encryption.*** Data encoding alone is not sufficient to provide complete confidentiality, but it could be enforced by using group keys to **partially encrypt** some of the coded chunks. Figure 5 shows a high-level view of the NDTP encryption operation. In the example, two consumers retrieving the same content have two different encoding methods but only one group key. Having a group key ensures that caches are not able to decrypt cached chunks. The group key also allows for overlapping requests to benefit from transparent caching.

The basic approach serves two goals. First, it provides privacy among consumers retrieving the same content by encoding part of the content differently for each consumer. Second, given that

each consumer has its own secret on how to decode part of the content, an adversary cannot easily infer if consumers are retrieving the same content. The reason why consumers with the same group key cannot be sure about other members of the same group is that chunks used in the data obfuscation might have been used to encode other content as well. This should be sufficient to prevent consumers from decoding these chunks, mainly when they have limited resources. Besides, by using the group key, we are guaranteeing **partial caching** for all the chunks that are encrypted using the group key.

To increase the level of obfuscation, the producer can apply an all-or-nothing transform, where caches or consumers in the same content list cannot decode the content unless all chunks are known. The trade-off with obfuscation and using group keys is that more coded chunks that combine the original content are needed to decode it. However, the older the chunks used to obfuscate the content, the higher the chance that it will be retrieved from a closed cache. Also, this method can be used to populate caches with chunks from other content objects.

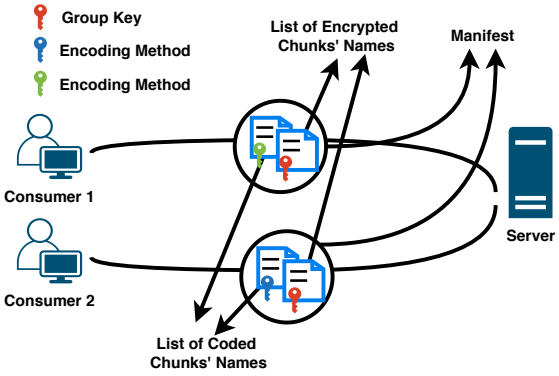


Figure 5: Partial encryption in NDT

5.2 Manifest Privacy

To provide partial encryption while also leveraging caching, only the manifest has to be secured using public key encryption. By using a specific URL format, an NDTP consumer knows that its request is for private content (like in HTTPS). In this case, there is no need to query the DNS server for the manifest record of the content. Instead, the consumer sends its request to the IP address of the content provider, just as in HTTPS. However, the NDTP consumer can still rely on the DNS to retrieve the necessary keys of the NDTP producer (server). This can be done by using techniques like DNS-based Authentication of Named Entities (DANE) [58], which allows the publication of Transport Layer Security (TLS) keys in zones for applications to use. After the NDTP consumer retrieves the IP address and the server key, it can issue a *Manifest Interest* encrypted using both the client and the server public and private key to ensure its authentication, integrity, and encryption. Once the server proves the client is authorized to request the content, it will then send back the manifest. While this is happening, NDTP caches on the way will not be able to intercept and understand either the Manifest Interest or the manifest itself since they are encrypted. To allow transparent caching while also ensuring privacy, the manifest contains a specific encoding method that is unique for each consumer and a group key that is unique for each content as we explained earlier.

5.3 Name Privacy and Access Control

Because Manifest Interests and manifests are encrypted using public-key encryption, no intermediate node is able to access any information inside them, including the name of the content requested. This means that consumers that are seeking the same content will not know who else is requesting it since both the Manifest Interest and the manifest are encrypted using the consumer and the producer keys. The same applies to members that are not part of the content group as well.

As we explained before, NDTP uses a chunk hash digest as its name. To prevent an adversary from reversing these chunk names to their original content, a one-way hash can be used. In generating these hash digests, NDTP producers use a randomly generated number (salt) to control access to chunk names from consumers who have their access revoked. This salt added to the manifest along with a timer on when this manifest and salt will expire. An NDTP consumer who was at some point authorized to receive a certain content but not anymore is unable to infer whether a particular chunk name will be related to a specific content object without the current salt value. Besides, consumers who have their access revoked will also be unable to request the content.

5.4 Multicast Support at the Transport Layer

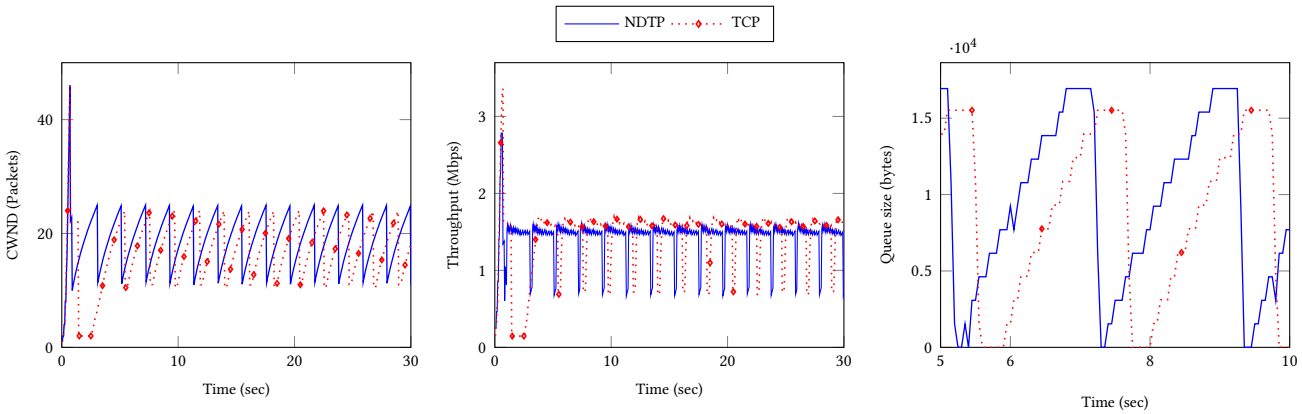
The IP multicast architecture in place today is based on the approach introduced by Deering and Cheriton [29]. The limitations of this approach have been discussed multiple times in the past [66], and among them are the need for global agreements on group addresses and the use of multicast routing protocols. CCN and NDN are able to provide "native multicast support," (i.e., supporting multicast delivery without multicast addresses and multicast routing protocols) by using PIT's to track pending Interests for multicast content denoted by name.

As we have explained, NDT proxies use PIT's to track pending Interests at the transport layer, and the aggregation of pending Interests at NDT proxies is very similar to that of NDN routers near consumers, but is based on IP addresses. However, this still leaves multiple copies of Interests for the same object flowing through the routers along the paths between NDT proxies at the edge and content sites. To reduce this traffic, network providers may choose to deploy layer-four switches to intercept and redirect NDT traffic to caching proxies at different network locations between popular mirrored content producers and customers.

The combined use of NDT proxies and manifest records with which consumers can be redirected to nearest mirroring sites results in similar functionality as a CDN, but without the need for overlays.

6 PERFORMANCE COMPARISON

We compared the performance of NDT, TCP, and NDN using our NDT implementation in ns-3 [97] and off-the-shelf implementations of TCP, DNS, and NDN in ns-3 and ndnSIM [77]. The ns-3 implementation of NDT is publicly available to the research community [4] to facilitate reproducibility of results and future NDT improvements.



(a) CWND Size (b) Throughput (c) Router's Buffer Occupancy

Figure 6: Single-flow scenario for NDTP and TCP

6.1 Efficiency of Congestion Control and Retransmission Mechanisms in NDTP

This experiment illustrates the inherent benefits of using a receiver-driven connection-free reliable transport protocol based on manifests instead of a connection-based transport protocol like TCP. We compared the congestion-control and retransmission mechanisms of NDTP and TCP assuming a scenario consisting of a simple network with a single source and a single sink. The topology of the network is a single path of four nodes with a single consumer/client at one end and a producer/server at the other end. Both ends share a common bottleneck of 1.5 Mbps and no in-network caching takes place. The propagation delay between the two ends is set to $40ms$. The consumer in NDTP issues Interests for the content served at the other end after requesting the manifest for this content from the producer. The client in TCP consumes traffic generated by the server after establishing a connection with it using the TCP three-way handshake. The size of the object chunks in NDTP is equal to the segment size in TCP, and fixed at 1500 bytes and both NDTP and TCP share the same fixed-header size.

Figure 6a shows the evolution of the congestion window for both protocols during the first $30s$ of the downloading a content of $3.69MB$, a total of 2465 chunks/segments. The growth of the congestion windows for both protocols matches the expected behavior of the additive-increase multiplicative-decrease (AIMD) algorithm.

The retransmission policy in NDTP allows receivers to detect and recover from a packet loss faster than in TCP, where it took the client a total of 35.5s to download the file compared to the total download time of 33.2s in NDTP. This is due to the fact that NDTP does not use connections and applies a fast retransmission strategy enabled by manifests. A consumer in NDTP has a complete picture of which OC's were received correctly and which were lost, and does not rely on partial ACK's like TCP does. Accordingly, the consumer immediately goes into congestion avoidance state, instead of fast recovery. As a result, NDTP continues increasing its congestion window normally. This allows NDTP to use the bottleneck's buffer more efficiently compared to TCP, which is forced into fast recovery, during which the sender can only transmit new data for every duplicate ACK received.

Figure 6c shows the queue size of the bottleneck's buffer for only 5 seconds of the simulation to highlight the idle periods of each protocol. It can be seen from the figure that TCP has more extended idle periods compared to NDTP. As a result, NDTP achieved higher average throughput due to better utilization of the link's capacity and the buffer size.

6.2 Efficiency of Transparent Caching

This experiment highlights the ability of NDT to take advantage of in-network caching like NDN does, but without requiring any changes to the IP routing infrastructure. We compared the total average time taken to retrieve multiple copies of a large data file using NDT, TCP, and NDN. The experiment consists of a source node connected over a 10 Mbps shared link to a cluster of six consumers, all interconnected via 100 Mbps links. An intermediate router is configured to forward NDT traffic to a caching proxy for NDT traffic. The same topology was used for NDN as well. Both NDN and NDT use the same congestion control algorithm, which mimics the TCP congestion control algorithm to provide a fair comparison with TCP. The scenario was run six times, and each time we increased the number of consumers in the network. All consumers start pulling a 6MB data file from the source simultaneously, and the total download time for every consumer is displayed in Figure 7.

As can be observed from Figure 7, TCP, NDT, and NDN perform very much the same when a single consumer is involved. This is to be expected, given that most of the NDT and NDN data packets are retrieved from the source in this case, and all three approaches use similar algorithms for congestion control. As the number of consumers increases, the completion times in NDT and NDN remain constant for all six scenarios. In contrast, the completion time in TCP increases linearly because all the data must be retrieved from the source. The use of PIT's in NDT and NDN results in only the first consumer Interests traversing the path to the producer, while the rest of the Interests are simply added to the PIT of the first router in NDN and the caching proxy in NDT.

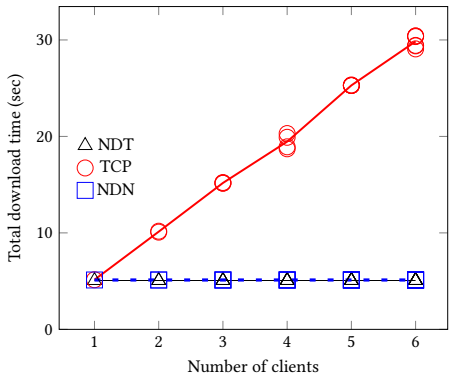


Figure 7: Transfer time vs. number of consumers

6.3 Impact of Manifest Records and Mirroring

This experiment highlights how NDT's architectural components work together to provide efficient name-based content delivery over the existing IP Internet. We compared the total average time taken to retrieve a large data file in three cases, namely: Using only the transparent caching enabled by NDT proxies (NP), using redirection to nearest mirroring sites based on my-DNS without transparent caching at NP's, and using NP's together with with redirection to nearest mirroring sits based on my-DNS.

Figure 8 shows the topology used in this scenario, which consists of multiple edge networks connected by a cluster of multiple consumers and mirroring sites located between the edge and the cloud network where the content server is located. Each edge router is connected to NDT proxies that provide transparent caching for NDT traffic passing through them. When my-DNS is enabled, Interests from consumers are routed to the nearest mirroring site for the content. Each experiment was run five times and the number of consumers in each cluster was increased. Each consumer starts pulling a 6MB data file from the producer at a random start time based on a Poisson distribution with a short average arrival time.

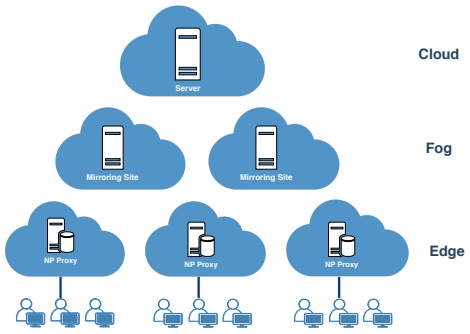


Figure 8: Network topology

Figure 9 shows the average latency incurred in retrieving the content object for each scenario, along with the variance. As expected, NDT performs its best when NDT proxies and nearest-replica routing through the my-DNS are used. As Figure 9 indicates, when only NP's are used, Interests from consumers have to reach the producer site, and the benefits of using NP's come from aggregating Interests and caching content. However, because of the short inter-arrival time of Interests, only aggregation is useful. In our scenario with ten consumers, only two Interests for the same data object traverse the path to the producer. Using my-DNS without NP's results in a shorter retrieval time for consumers, but duplicate packets are sent along links. When both my-DNS and NP's are used, Interests from consumers are routed to the nearest mirroring site and aggregated at the NP's.

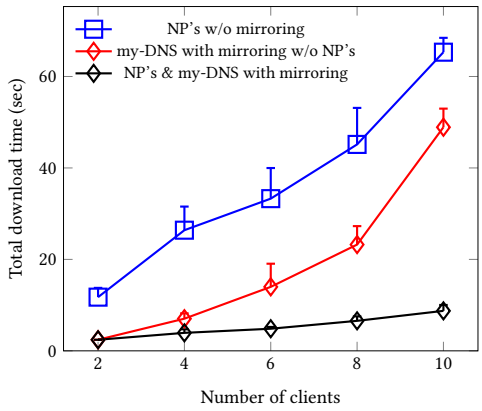


Figure 9: Total download time for different approaches to the use of mirroring and caching in NDT

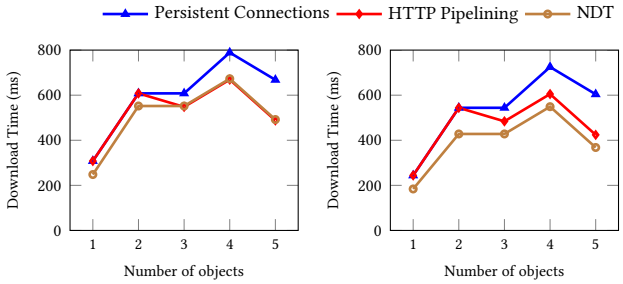
6.4 Overhead of URL to Manifest Mapping

This experiment illustrates the overhead of mapping URLs to manifest records using my-DNS. The scenario is based on the HTTP application, where clients start requesting the main web page and then start requesting the embedded inline objects based on their URL in the web page. The main object size and the size of the embedded inline objects are based on the top one million visited web pages indicated in [84]. We used two HTTP applications for our comparison, one based on persistent connections, in which a new HTTP request cannot be sent until the response to the current request is received. The other application is based on HTTP pipelining, where multiple HTTP requests can be sent together over a single TCP connection. For the case of NDT, each URL mapped to a single manifest record. The NDTP consumer starts querying the my-DNS for the manifest record of the main object, and it queries for the inline objects records after retrieving the main object from the server.

The topology of the network is a single path of four nodes with a single client connected to its local my-DNS server, and a content server at the other end that is connected to its authoritative my-DNS server. For the sake of simplicity, the IP address of the authoritative my-DNS server is cached at the local my-DNS server at the start of the simulation. Both resource and manifest records need to be retrieved from the authoritative my-DNS server at the other end if they are not cached. For a fair comparison between NDT and HTTP over TCP, NDT and TCP have the same fixed-header size, the same chunk and segment size, and the same congestion control algorithm.

As Figure 10a shows, NDT performs at least as well as HTTP pipelining. Both HTTP applications require a connection to be established using TCP three-way handshakes before a client can start sending and receiving data. By contrast, NDTP allows clients to retrieve data without the need to establish a connection, which reduces the number of RTTs by at least one compared to TCP. Multiplexing is easily supported in NDTP because objects in NDTP are globally named and pointed by their own manifest, allowing consumers to pipeline and multiplex multiple objects together. As seen in Figure 10b, when manifest records are cached, NDT outperformed both types of HTTP. This proves that using my-DNS to translate URLs as structured in applications like HTTP do not impose significant overhead in NDT.

Using my-DNS in NDT does not impose significant overhead compared to NDN. A consumer in NDN has to retrieve the manifest from the producer before issuing Interests to retrieve the content. Retrieving the manifest record using my-DNS adds only the additional delay incurred in redirecting the consumer to the site with the manifest.



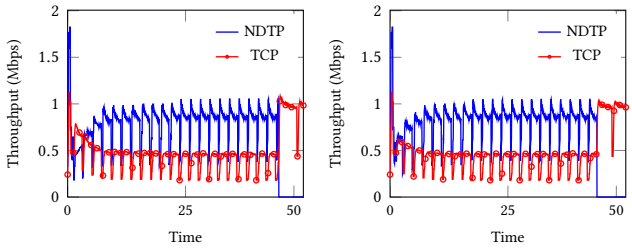
(a) No cached DNS records (b) With cached DNS records

Figure 10: Overhead of URL-to-manifest mapping

6.5 TCP Friendliness

This experiment is used to illustrate TCP friendliness in NDTP with and without caching. The topology of the network consists of a bottleneck link of capacity 1 Mbps and a buffer size of 20 packets. For the sake of simplicity, the chunk size of NDTP is fixed at 1000 bytes, and the same goes for the segment size of TCP. TCP operates with the SACK option enabled, and ACK's are not delayed. Both TCP and NDTP have the same round-trip-time delay, and they are retrieving the same file of about 3MB. For the scenario with cashing, NDT caching proxies are configured in the topology before the bottleneck. This allows the NDT caching proxy to serve the consumer Interests for dropped data packets due to congestion along the bottleneck. Initially, the NDT caching proxy is empty and caches any data packets that pass through it. A router is configured to interrupt NDTP packets based on the protocol number and redirect them to the NDT caching proxy.

Figure 11a shows the results with and without caching. Using Jain's fairness index, the fairness between the two flows without caching is equal to $F = 0.9988$, which is understandable because NDTP also follows an AIMD congestion control algorithm like TCP. Similar results occur when caching takes place. The results illustrate that caching does not have a large negative effect on fairness. In fact, fairness between the two flows was equal to $F = 0.9967$ with caching. This is due to the ability of NDTP to detect that most packets were retrieved from the primary source, and control its sending rate accordingly. Even though NDTP achieved less fairness than two competing TCP flows under the same scenario (where fairness equals $F = 0.999996$), the total download time for TCP flows retrieving the same file size was higher by 8.5%.



(a) Without caching (b) With caching

Figure 11: TCP Friendliness

7 CONCLUSIONS AND FUTURE WORK

We introduced Named-Data Transport (NDT), the first ICN approach that attains efficient content dissemination without end-to-end connections or modifications to the IP routing infrastructure. The design of NDT provides the same benefits of NDN and similar ICN architectures through the integration of a new connectionless reliable transport protocol with name resolution and NDT proxies that support privacy-preserving on-line caching and native multicasting.

The results of simulation experiments in ns-3 show that: (a) NDT is inherently more efficient than TCP, (b) the performance of NDT and NDN is very similar, and (c) NDT outperforms HTTP over TCP while being able to provide privacy.

We implemented congestion and retransmission control algorithms in NDT that are similar to those used in TCP simply to highlight the inherent benefits of the name-based connectionless approach used in NDTP. Far more efficient algorithms can be adapted to be used in NDTP, including many that have been proposed for TCP recently [16, 17], and this is an area of future work. Similarly, our initial design of mechanisms to secure content and manifest records should be viewed as a starting point, and clearly more work is needed in this area. Making native multicast more efficient in NDT is another area of future work.

We focused on static content in our discussion of NDT; however, the approaches that have been described for the support of real-time voice and video-conferencing in NDN and CCN [55, 61] are equally applicable to the end-to-end information-centric approach in NDT.

The implementation of NDT in ns3 is publicly available [4] to foster further research on NDT and similar information-centric Internet solutions that preserve and evolve with the IP routing infrastructure.

REFERENCES

[1] E.G. AbdAllah, H.S. Hassanein, and M. Zulkernine, "A Survey of Security Attacks in Information-Centric Networking," *IEEE Communications Surveys & Tutorials*, 2015.

[2] B. Ahlgren et al., "A Survey of Information-Centric Networking," *IEEE Communications Magazine*, July 2012, pp. 26–36.

[3] A. Albalawi and J. J. Garcia-Luna-Aceves, "A Delay-Based Congestion-Control Protocol for Information-Centric Networks," *Proc. IEEE ICNC '19*, 2019.

[4] A. Albalawi and J.J. Garcia-Luna-Aceves, "NDT ns-3 Simulator." Available at: https://github.com/aalbalaw/NDT.

[5] M. Ambrosin et al., "Security and Privacy Analysis of National Science Foundation Future Internet Architectures," *IEEE Communications Surveys & Tutorials*, 2018.

[6] T. Anderson et al., "A Brief Overview of the NEBULA Future Internet Architecture," *ACM SIGCOMM CCR*, 2014.

[7] S. Arianfar et al., "On Preserving Privacy in Content-Oriented Networks," *Proc. ACM SIGCOMM Workshop on Information-Centric Networking*, 2011.

[8] S. Ariyapperuma and C. Mitchell, "Security vulnerabilities in DNS and DNSSEC," *Proc. IEEE ARES '07*, April 2007.

[9] R. Atkinson, S. Bhatti, and S. Hailes. "ILNP: Mobility, Multi-homing, Localised Addressing and Security through Naming," *Telecommunication Systems*, 2009.

[10] P. Baran, "On Distributed Communications Networks," *IEEE Transactions on Communications Systems*, March 1964.

[11] M.F. Bari et al., "A Survey of Naming and Routing in Information-Centric Networks," *IEEE Communications Magazine*, July 2012, pp. 44–53.

[12] N. Bartolini, E. Casalicchio, and S. Tucc, "A Walk through Content Delivery Networks," *Proc. IEEE MASCOTS '03*, 2003.

[13] M. Baugher et al., "Self-Verifying Names for Read-Only Named Data," *Proc. IEEE INFOCOM Workshops '12*, March 2012.

[14] BitTorrent. http://bittorrent.com

[15] M.S. Blumenthal and D.D. Clark, "Rethinking the Design of the Internet: The End-to-End Arguments vs. the Brave New World," *ACM Trans. on Internet Technology*, Aug. 2001.

[16] N. Cardwell et al., "BBR: Congestion-Based Congestion Control," *ACM Queue*, Oct. 2016.

[17] N. Cardwell et al., "Model-based Network Congestion Control," *Technical Disclosure Commons*, March 27, 2019.

[18] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking," *Proc. IEEE NOMEN '12*, March 2012.

[19] G. Carofiglio et al., "Enabling ICN in the Internet Protocol: Analysis and Evaluation of the Hybrid-ICN Architecture," *Proc. ACM ICN '19*, Sept. 2019.

[20] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, " Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service," *Proc. ACM PODC '20*, 2000.

[21] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, "Content-based Addressing and Routing: A General Model and Its Application," Tech. Report CU-CS-902-00, Univ. of Colorado, Jan. 2000.

[22] A. Carzaniga and A.L. Wolf, "Content-Based Networking: A New Communication Infrastructure," *Proc. Workshop on Infrastructure for Mobile and Wireless Systems*, 2002.

[23] V.G. Cerf, Y.K. Dalal, and C.A. Sunshine, "Specification of Internet Transmission Control Program," INWG Note 72, revised Dec. 1974.

[24] Q. Chen et al., "Transport Control Strategies in Named Data Networking: A Survey," *IEEE Communications Surveys & Tutorials*, 2016.

[25] X. Chen et al., "Maintaining Strong Cache Consistency for the Domain Name System," *IEEE Transactions on Knowledge and Data Engineering*, August 2007

[26] S. Cheshire, J. Graessley, and R. McGuire, "Encapsulation of TCP and other Transport Protocols over UDP," Internet Draft, July 2013.

[27] A. Dabirmoghaddam, M.M. Barijough, and J. J. Garcia-Luna-Aceves, "Understanding Optimal Caching and Opportunistic Caching at the Edge of Information-Centric Networks," *Proc. ACM ICN '14*, Sept. 2014.

[28] A. Dabirmoghaddam, M. Dehghan, and J.J. Garcia-Luna-Aceves, "Characterizing Interest Aggregation in Content-Centric Networks," *Proc. IFIP Networking 2016*, May 17–19, 2016.

[29] S.E. Deering and D.R. Cheriton, "Multicast Routing in Datagram Internetwork and Extended LANs," *ACM TOCS*, May 1990.

[30] E. Demirors and C. Westphal, "DNS++: A Manifest Architecture for Enhanced Content-Based Traffic Engineering," *Proc. IEEE GLOBECOM '17*, 2017.

[31] T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," 2008.

[32] D. E. Eastlake 3rd, "Domain Name System Security Extensions," RFC 2535, 1999.

[33] W.M.Eddy, "At What layer Does Mobility Belong?," *IEEE Communications Magazine*, 2004.

[34] A. Eriksson and A. Mohammad Malik, "A DNS-Based Information-Centric Network Architecture Open to Multiple Protocols for Transfer of Data Objects, *Proc. IEEE ICIN '18*, 2018.

[35] S. K. Fayazbakhsh et al., "Less Pain, Most of the Gain: Incrementally Deployable ICN," *Proc. ACM SIGCOMM '13*, 2013.

[36] D. Florez-Rodriguez et al., "Global Architecture of the COMET System," Seventh Framework STREP No. 248784, 2013.

[37] FP7 COMET project. [Online]. Available: http://www.comet-project.org/

[38] FP7 PSIRP project. [Online]. Available: http://www.psirp.org/

[39] FP7 PURSUIT project. [Online]. Available: http://www.fp7-pursuit.eu/PursuitWeb/

[40] FP7 SAIL project. [Online]. Available: http://www.sail-project.eu/

[41] FP7 4WARD project. [Online]. Available: http://www.4ward-project.eu/

[42] FP7 CONVERGENCE project. [Online]. Available: http://www.ictconvergence.eu/

[43] Z. Gao, A. Venkataramani, and J.F. Kurose, "Towards a Quantitative Comparison of Location-Independent Network Architectures," *ACM SIGCOMM CCR*, 2014.

[44] J.J. Garcia-Luna-Aceves, "System and Method for Discovering Information Objects and Information Object Repositories in Computer Networks," U.S. Patent 7,162,539, January 9, 2007.

[45] J. J. Garcia-Luna-Aceves, "Name-Based Content Routing in Information Centric Networks Using Distance Information," *Proc. ACM ICN '14*, Sept. 2014.

[46] J.J. Garcia-Luna-Aceves, Q. Li, and Turhan Karadeniz, "CORD: Content Oriented Routing with Directories," *Proc. IEEE ICNC '15*, Feb. 2015.

[47] J.J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, "Content-Centric Networking Using Anonymous Datagrams," *Proc. IFIP Networking '16*, May 2016.

[48] J.J. Garcia-Luna-Aceves, M. Mirzazad-Barijough, and E. Hemmati, "Content-Centric Networking at Internet Scale through The Integration of Name Resolution and Routing," *Proc. ACM ICN '16*, Kyoto, Japan, Sept. 2016.

[49] J.J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, "Efficient Multicasting in Content-Centric Networks Using Datagrams," *IEEE Globecom '16*, Dec. 2016.

[50] J.J. Garcia-Luna-Aceves, "New Directions in Content Centric Networking,"*Proc. IEEE CCN '15* Oct. 2015.

[51] C. Ghasemi et al., "MUCA: New Routing for Named Data Networking," *Proc. IFIP Networking '18*, May 2018.

[52] D.K. Gifford, "Replica Routing," U.S. Patent 6,052,718, April 18, 2000.

[53] M. Gritter and D. Cheriton, "An Architecture for Content Routing Support in The Internet," *Proc. USENIX Symposium on Internet Technologies and Systems*, Sept. 2001.

[54] Y. Gu and R. Grossman, " UDT: UDP-Based Data Transfer for High-Speed Wide Area Networks," *Computer Networks*, Elsevier, 2007.

[55] P. Gusev and J. Burke, "NDN-RTC: Real-Time Videoconferencing over Named Data Networking," *Proc. ACM ICN '15*, Sept. 2015.

[56] D. Han et al., "XIA: Efficient Support for Evolvable Internetworking," *Proc. USENIX NSDI '12*, 2012.

[57] E. Hemmati and J.J. Garcia-Luna-Aceves, "A New Approach to Name-Based Link-State Routing for Information-Centric Networks," *Proc. ACM ICN '15*, Sept. 2015.

[58] P. Hoffman and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA," RFC 6698, 2012.

[59] V. Jacobson, "Congestion Avoidance and Control, *Proc. ACM SIGCOMM '88*, Aug. 1988.

[60] V. Jacobson et al., "Networking Named Content," *Proc. ACM CoNEXT '09*, Dec. 2009.

[61] V. Jacobson et al., "VoCCN: Voice-over Content-Centric Networks," *Proc. ACM ReArch '09*, Dec. 2009.

[62] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, IETF, March 2006.

[63] T. Koponen et al., "A Data-Oriented (and Beyond) Network Architecture," *Proc. ACM SIGCOMM '07*, Aug. 2007.

[64] A. Langley et al., "The QUIC Transport Protocol: Design and Internet-Scale Deployment," *Proc. ACM SIGCOMM '17*, Aug. 2017.

[65] D. Le, X. Fu, and D. Hogrefe, "A Review of Mobility Support Paradigms for the Internet, *IEEE Communications Surveys & Tutorials*, 2006.

[66] B.N. Levine et al., "Consideration of Receiver Interest for IP Multicast Delivery," *Proc. IEEE Infocom '00*, March 2000.

[67] J. Li, "On peer-to-peer (P2P) content delivery," *Peer-to-Peer Netw. Appl.*, 2008.

[68] E.K. Lua et al., "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," *IEEE Communications Survey & Tutorial*, March 2004.

[69] N.A. Lynch, Y. Mansour, and A. Fekete, "Data link layer: Two Impossibility Results," *Proc. ACM PODC '88*, 1988.

[70] N.A. Lynch, *Distributed Algorithms*, Morgan Kauffman, 1996.

[71] P. Mockapetris, "Domain Names – Implementation and Specification," RFC 1035, Nov. 1987.

[72] I. Moiseenko and L. Zhang. "Consumer-producer API for Named Data Networking," *Proc. ACM ICN '14*, 2014.

[73] I. Moiseenko, " Fetching content in Named Data Networking with Embedded Manifests," 2014.

[74] M. Mosko, I. Solis, E. Uzun, C. Wood, "CCNx 1.0 Protocol Architecture," Xerox PARC, April 2017.

[75] L. Muscariello et al., "Hybrid Information-Centric Networking," IETF Internet Draft, Oct. 2019.

[76] NSF Named Data Networking project. [Online]. Available: http://www.named-data.net/

[77] NS-3 based Named Data Networking (NDN) simulator [Online]. Available: https://ndnsim.net/current/index.html
[78] NSF Mobility First project. [Online]. Available:http://mobilityfirst.winlab.rutgers.edu/.
[79] E. Nygren, R.K. Sitaraman, and J. Sun, "The Akamai Network: A Platform for High-Performance Internet Applications," *ACM SIGOPS Operating Systems Review*, Aug. 2010.
[80] G. Papastergiou et al., "De-ossifying the Internet Transport Layer: A Survey and Future Perspectives," *IEEE Communications Surveys & Tutorials*, Nov. 2016.
[81] R. Peon, "Explicit proxies for HTTP/2.0," IETF Informational Internet Draft, 2012.
[82] E. Perera, V. Sivaraman, and A. Seneviratne, "Survey on Network Mobility Support," *ACM SIGMOBILE Mobile Computing and Communications Review*, 2004.
[83] M. Polese et al., "A Survey on Recent Advances in Transport Layer Protocols," *IEEE Communications Surveys & Tutorials*, Aug. 2019.
[84] R. Pries, Z. Magyari, and P. Tran-Gia, "An HTTP Web Traffic Model based on the Top One Million Visited Web Pages," *Proc. IEEE Euro-NF Conference on Next Generation Internet '12*, 2012.
[85] J. Raju, J.J. Garcia-Luna-Aceves, and B. Smith, "System and Method for Information Object Routing in Computer Networks," U.S. Patent 7,552,233, June 23, 2009.
[86] D. Saha et al., "Mobility Support in IP: A Survey of Related Protocols," *IEEE Network*, 2004
[87] J.H. Saltzer, "End-to-End Arguments in System Design," RFC 185, 1980.
[88] L. Saino, C. Cocora, and G. Pavlou, "CCTCP: A Scalable Receiver-Driven Congestion Control Protocol for Content Centric Networking," *Proc. IEEE ICC '13*, 2013.
[89] I. Seskar et al., "MobilityFirst Future Internet Architecture Project," *Proc. AINTEC '11*, Nov. 2011.
[90] S. Sevilla, P. Mahadevan, and J.J. Garcia-Luna-Aceves, "iDNS: Enabling Information Centric Networking through the DNS," *Proc. IEEE INFOCOM Workshop on Name-Oriented Mobility '14*, 2014.
[91] S. Sevilla and J.J. Garcia-Luna-Aceves, "Freeing The IP Internet Architecture from Fixed IP Addresses," *Proc. IEEE ICNP '15*, Nov. 2015.
[92] S. Sevilla, J.J. Garcia-Luna-Aceves, and H. Sadjadpour, "GroupSec: A New Security Model for the Web," *Proc. IEEE ICC '17*, 2017.
[93] S. Sevilla and J.J. Garcia-Luna-Aceves, "A Deployable Identifier-Locator Split Architecture," *Proc. IFIP Networking '17*, June 2017.
[94] J.M. Spinelli, "Reliable Communication on Data Links," LIDS-P-1844, MIT, Dec. 1988.
[95] A. Stubblefield and D. Wallach, "Dagster: Censorship-Resistant Publishing Without Replication," Rice University, Dept. of Computer Science, Tech. Rep. TR01-380, 2001.
[96] B. Tremblay et al., "(D.A.3) Final Harmonised SAIL Architecture," Report FP7-ICT-2009-5-257448-SAIL/D-2.3, Feb. 2013.
[97] F. Urbani, W. Dabbous, and A. Legout, NS3 DCE CCNx quick start, INRIA, Nov. 2011.
[98] P. Vixie et al., "Dynamic Updates in the Domain Name System," IETF RFC 2136, 1997.
[99] L. Wang et al., "A Secure Link State Routing Protocol for NDN," *IEEE Access*, March 2018.
[100] C. Westphal and E. Demirors, "An IP-Based Manifest Architecture for ICN," *ACM ICN Demo*, September 2015.
[101] Y. Wu, J. Tuononen, and M. Latvala, "Performance Analysis of DNS with TTL Value 0 as Location Repository in Mobile Internet," *IEEE WCNC '07*, March 2007.
[102] G. Xylomenos et al., "Caching and Mobility Support in a Publish-Subscribe Internet Architecture," *IEEE Communications Magazine*, July 2012
[103] G. Xylomenos et al., "A Survey of Information-centric Networking Research," *IEEE Communication Surveys & Tutorials*, July 2013.
[104] B. Zolfaghari et al., "Content Delivery Networks: State of the Art, Trends, and Future Roadmap," *ACM Computing Surveys*, April 2020.