Exploring Named Data Networking and its Security Considerations on FABRIC

Alexander Gaouette
Arizona State University
agaouett@asu.edu

Brandon McMillin
Arizona State University
brmcmil2@asu.edu

***Abstract*—Named Data Networking (NDN) is a project with intent on changing how the current IP architecture looks and works. The current IP architecture is in need of redesign as it was built with a vision to work as a communication network, but the Internet has grown to be more of a distribution network. With this new proposed architecture, questions have arisen as to how security and trust will be implemented. We are using the FABRIC testbed to implement the NDN architecture, as well as to implement and integrate Name-Based Access Control (NAC). In future work, we additionally seek to implement the NDN Project's trust schema specification for automated data and interest packet signing and authentication. Our goal is to capture different network metrics, such as packet processing time and RTT, and show that the addition of such security layers to NDN provide significant value with little to no downside.**

***Index Terms*—Named Data Networking, FABRIC, Network Security, Access Control, Trust**

I. Introduction and Motivation

Named Data Networking (NDN) is a project that has a purpose in redesigning the network architecture of today's Internet. IP, the network layer communications protocol, was designed to create a communication network [1], but growth in many areas of the internet such as social media, e-commerce, and digital media has led to the Internet being used more as a distribution network. With that in mind, NDN seeks to change the way people access content over the Internet, ultimately replacing the narrow waist of the Internet that is the IP layer. This is accomplished by fetching content chunks that are identified by a name, rather than the current implementation where packets are requested from some destination address and sent back to the source.

The opportunity to reshape the current Internet architecture also allows for the conception of a more security-conscious construction. Compared to the current protocol stack, NDN is security-centric by design [1]. Trust is based on name in NDN, where the validity of a data packet is determined by whether it is signed by a key that satisfies certain conditions [2]. This allows for novel content access control methods to be implemented directly as is the case with Name-based Access Control (NAC) [4], as well as differing trust schema specifications that have been presented as ways to help secure the NDN architecture. In an effort to gain a better understanding of both NDN and its security considerations, our goal is to build upon an existing NDN implementation [7] that is deployable on the FABRIC testbed and the integration of the NAC model. In future work, we also aim to implement the trust schema described by the Named Data Networking Project specifications, though this will likely be reserved for future work. Finally, we hope to demonstrate that the addition of one, or both of these features does not add a significant overhead in network performance by evaluating metrics such as packet processing time.

II. Related Work

A. Named Data Networking

In [1], the authors of the paper propose a different internet architecture known as Named Data Networking (NDN). NDN is said to "change the semantics of network service from delivering the packet to a given destination address to fetching data identified by a given name." This is important as the current Internet was designed as a communication network, and now it is being used as much more. In the proposed architecture of NDN, an NDN router needs to maintain three data structures: a Pending Interest Table (PIT), a Forwarding Information Base (FIB), and a Content Store (CS), as well as a Forwarding Strategy module. The structures and module hold all the information in the router that is needed to get the consumer the data they have requested. What makes NDN so unique is that data is fetched by name, and each application is allowed to choose the naming scheme that fits its needs best. In addition, it has in-network storage where the CS can help shorten round trip time by having data a user may want already cached in network at the router.

B. Name-Based Access Control of NDN

In Name-Based Access Control [4], the authors explain that NAC is a content-based access control model that enforces access control directly over content associated with a given namespace through the encryption of that content at the time of production. The owner of the data is able to limit the production rights of other data producers seeking to produce content within the owner's namespace, and the access rights of data consumers attempting to read said content. This is accomplished through the combined use of both public-key and symmetric key cryptography as well as signing certificates. The results from the paper suggest that NAC is better for larger scale distributed data production and consumption. As this subsection pertains to a significant portion of the scope of

our project, it will be expanded with more relevant background information.

C. Trust Schema of NDN

The proposed trust schemas for NDN are a description of a trust model that automates the process of interest packet signing, authentication [2]. The authors of [2] and [3] explain that a trust schema consists of three parts: a list of rules, one or more trust anchors, and a cryptography specification. The rules are restrictions on packet names as well as its signing key name, the anchor is a data packet carrying the public key, and the cryptography specification gives requirements on packet signature as to which public key algorithm to use, hashing algorithm to use, and the minimum required signature strength. Different public key signing algorithms, such as RSA and ECDSA, can affect the performance of trust schema. [10] ECDSA has been shown to be better than RSA due to the running time in both key generation and signing being faster, but RSA does have a quicker signature verification.

III. Experimental Set-up

A. Establishing the Baseline NDN Architecture

The first phase of the project was largely concerned with the initial, baseline deployment of the NDN architecture onto the FABRIC testbed. To simplify matters as to better focus on the security features we aim to examine, we are using an existing implementation of NDN on Fabric [7], produced by graduate students at Arizona State University, that streamlines the deployment of the NDN-DPDK project [5, 6]. As NDN-DPDK is a high-speed suite of programs that provide a multitude of tools such as a traffic generator, packet forwarder, and file server, but no current implementation of NAC or the NDN trust schema specification, it provides the perfect foundation for our project.

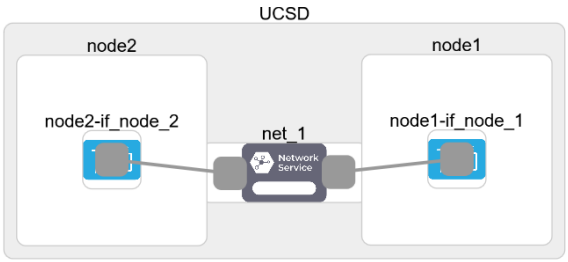

Fig. 1. Network Topology on FABRIC

As shown in Figure 1, the topology for our project consists of two nodes, each established as forwarding devices using NDN-DPDK, connected by a single L2 bridge and deployed on the UCSD FABRIC site. This allowed us to ensure proper functionality of the implementation before proceeding further. In the future, we plan to expand to a larger topology to better represent a more realistic network environment. With either topology, however, our intention was to collect baseline performance metrics such as data and interest packet processing time which can later be compared against metrics collected with NAC and the trust schema implemented. Due to implementation limitations, we were only able to determine Round Trip Time via observation of an NDNPing operation.

B. Name-Based Access Control Implementation

We aim to present a functional demonstration of the simplest form of NAC: encryption-based read access, based on a NAC presentation at ACM ICN 2015 [9] and its associated paper [8], includes the following:

1) Creation of a simple hierarchical namespace, where producers can publish produced content with authorization, and privileged users are provided read access to this content.
2) Implementation of production credentials. On the producer side, a symmetric content key is used to encrypt the data that will be accessed by the consumer.
3) Implementation of consumer credentials. In this form of NAC, consumers use public-key cryptography and share their public key with producers. The producer then encrypts the content key with the consumer's public key. Upon recipt of the encrypted content key, the consumer is able to utilize their private key to decrypt the content key, ultimately allowing the decryption (and thus read access) of the content.

Figure 2 below presents a graphical representation of this access control pipeline.

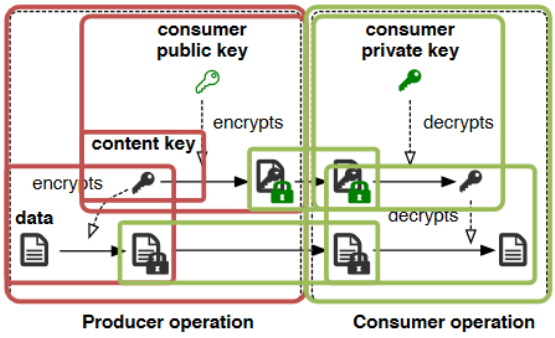

Fig. 2. NAC Encryption-Based Read Access. Retrieved from [9]

For this phase of the project, our intention was initially to integrate the NDN Project's NAC Library for NDN [8] with the NDN-DPDK implementation. An alternative to this design was comprehensive modification of the NDN-DPDK presented many difficulties over the course of this endeavor, however. Instead, two python applications, representing a consumer and a producer, were developed that act as wrappers to interact with the NDN-DPDK demonstration tools. This allowed us to more easily interact with the NDN-DPDK platform and still utilize the NDN architecture.

The producer application is instantiated on Node 2 in our topology. This application first publishes an entry to its FIB, and starts a process to monitor incoming interests utilizing

the program 'ndndpdk-godemo pingserver' from the platform. In another process, the producer launches a server instance to host to publish a file corresponding to the interest entry. This process similarly invokes the program 'ndndpk-godemo put'. The producer application then waits for interest packets from the sole consumer it is connected to. Upon receipt of interest, the process monitoring the interests retrieves the consumer's RSA public key directly from the consumer who is hosting the key on a server of their own. The content key stored on the producer's side is encrypted with the consumer's public key and published within the namespace, available for the consumer's future retrieval. In the current state of development, the identity of the consumer is known, but the possibility of multiple consumers would require restructuring of the producer application to support multiple clients.

The client application is similar in design. Upon start of the consumer application, a process is created to host the consumer's public key and an entry is added to the FIB on the configured interface. In another process, the consumer creates an interest for the file hosted by the producer and broadcasts interest packets. Upon interest reply, the consumer initiates a download of the producer's file. Once the file has been successfully retrieved, the consumer will then publish an interest for the encrypted content key and then retrieve it from the producer's namespace. The content key is then decrypted within the application using the consumer's private key. Figure 3 presents a diagram outlining the interactions between the two networked applications.

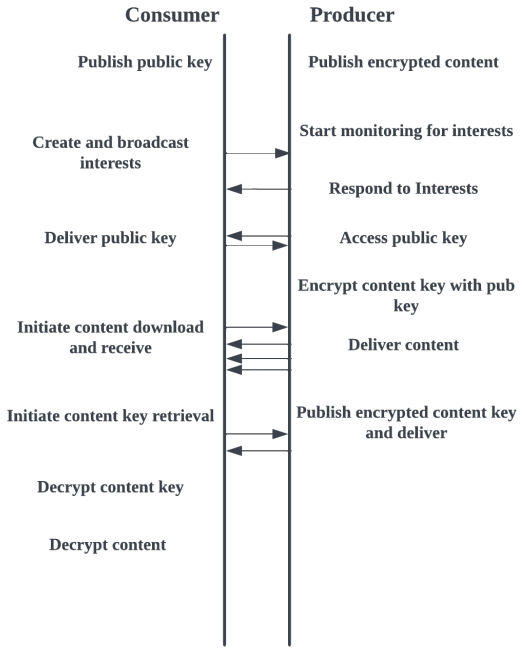

Fig. 3. Network Time Space Diagram of Program Interaction

Alongside the two applications, the project's Jupyter notebook builds the directories and files used within the demonstration. For the consumer, this includes generation of the public and private keys. The Jupyter notebook also contains cells that download and install all required dependencies for the project, configure the interfaces for both nodes, and instantiates the forwarders on them.

```
ubuntu@dd5790f8-26d5-4806-b2d9-efe12ccaa388-node1:~$ python consumer.py
Obtaining local face ID for Consumer Node.
Face ID: UJ104HBG8V52BG5I

Adding entry for /bob/blog/may04 to FIB on face UJ104HBG8V52BG5I.
Adding entry for /alice/access/alice_key.pub to FIB on face UJ104HBG8V52BG5I.
Publishing public key found at /home/ubuntu/alice/access/alice_key.pub to namespace /alice/access/alice_key.pub.

2023/05/04 08:44:54 uplink opened, state is down
2023/05/04 08:44:54 uplink state changes to up
Awaiting response to interest from server...

Interest packet reply received.
Adding entry for /bob/blog/may04.enc to FIB on face UJ104HBG8V52BG5I.

Attempting to download file.
File retrieved. Saving to /home/ubuntu/alice/downloads/may04.enc...
Encrypted file saved.

Getting content key...
Adding entry for /bob/blog/c_key_for_alice to FIB on face UJ104HBG8V52BG5I.

Attempting to download content key.
File retrieved. Saving to /home/ubuntu/alice/downloads/c_key_for_alice...
File saved.
Decrypting content key with private key.

Content key decryption successful. Saved to /home/ubuntu/alice/downloads/c_key
File and content key retrieved. Press Ctrl+C to exit.
```

Fig. 4. Full Execution of Consumer Application

IV. Results

Baseline NDN architecture implementation is fully functional and easily instantiated based on the work of [7]. Our topology, as specified in Section 3, consists of two nodes linked together by an L2 bridge. Both nodes are configured as packet forwarders, and we have observed the capability for communication between nodes by use of a simple ndnping. We are capable of collecting RTT measurements trivially in this way.

Over the course of developing this project several challenges and obstacles arose, requiring shifts in implementation design choices. Our initial goal was to utilize the NDN Projcet's NAC Library, but this library is fundamentally incompatible with NDN-DPDK. An implementation requiring the modification and extension of the NDN-DPDK platform was additionally considered, but presented a significant technical challenge. Ultimately the decision was made to prototype the NAC mechanisms into python applications whose subprocesses interact directly with the NDN-DPDK software for interest and data packet communication. The automation of the access control pipeline is fully realized and demonstrated by the communication between the consumer and producer applications, however. Figures 4 and 5 provide an example of successful access control authorization.

The constraints of the NAC implementation unfortunately leaves deeper, more meaningful forms of packet inspection and network measurements inaccessible. In the NAC demonstration's current state, the RTT of interest packets remain the sole metric the design is capable of capturing. While the transmission of packets is invoked by the applications, network performance is not impacted by the access control operations the processes execute. As a result, there are unfortunately no tangible metrics to analyze for comparison.

```
ubuntu@bf02a541-94b0-4899-b7f1-8209ae6137da-node2:~$ python producer.py
Obtaining local face ID for Producer Node.
Face ID: GO560IH8K6VFLDO

Adding entry for /bob/blog/may04 for discovery on GO560IH8K6VFLDO.
Adding entry for /bob/blog/may04.enc to FIB on face GO560IH8K6VFLDO.
Publishing file /home/ubuntu/bob/blog/may04.enc to server as /bob/blog/may04.enc.

Monitoring for incoming interest packets for /bob/blog/may04.
2023/05/04 08:44:40 uplink opened, state is down
2023/05/04 08:44:40 uplink state changes to up
2023/05/04 08:44:54 /8=bob/8=blog/8=may04/8=C9F1C2414275AB57[F]
Interest packet received.
Retrieving public key for user.
Key retrieved. Encrypting content key with user's public key...

Hosting content key for consumer retrieval.
Publishing file /home/ubuntu/bob/blog/c_key_for_alice to server as /bob/blog/c_key_for_alice.
```

Fig. 5. Full Execution of Producer Application

V. Summary, Conclusions, and Future Work

The goal of this project was to implement the NDN architecture, Name-Based Access Control, and the NDN Project's Trust Schema specification on the FABRIC testbed, and evaluate the network performance the added security adds. The NDN architecture was designed with intent to shift from the current host-centric network architecture (IP) to a more appropriate data-centric network architecture (NDN). As today's Internet is more commonly used as a distribution network as opposed to the communication network it was designed to be, NDN affords many benefits that better fit the needs of the modern Internet. Unburdened by the constraints of the past, NDN's novel architecture also presents the opportunity to make more security-conscious choices by design, rather than an afterthought. One of those choices, NAC, provides secure, distributed content sharing through automated content encryption directly at the time of publishing. In this way, content producers can better control access to content published on their networks.

Despite unforeseen incompatibilities and many technical challenges, we are able to present a functional demonstration of name-based access control. Though limited in its capabilities (e.g. handling multiple consumers and group policies), it realizes the fundamental design considerations published in [4]. The results of this project are fully reproducible on the FABRIC testbed via the Jupyter notebook published alongside the program source code on github [11]. Reproducibility of previous public work was critical in the development of this project. While the initial scope of the project was perhaps overly ambitious, we hope this project can provide a stepping stone towards the exploration of further NDN-based security considerations for others.

In future work, we would like to revisit a more robust, Go-based design of NAC that is fully integrated into the NDN-DPDK platform. Such an implementation would allow for more realistic network topologies and provide better insight into the performance overhead it brings with it. Furthermore, trust schema implementation provides an additional avenue for future security-based research into NDN. Determining the trustworthiness of a given piece of published content and making an informed decision on the content's legitimacy is perhaps more critical than ever given the ever increasing prevalence of misinformation disseminated on the Internet.

References

[1] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. 2014. Named data networking. SIGCOMM Comput. Commun. Rev. 44, 3 (July 2014), 66–73.

[2] NDN Trust Schema Specification [https://github.com/named-data/ndn-cxx/blob/feature-schema/docs/tutorials/security-trust-schema.rst]

[3] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. 2015. Schematizing Trust in Named Data Networking. In Proc. of ACM ICN.

[4] Yingdi Yu, Alexander Afanasyev, and Lixia Zhang. 2016. Name-based access control. Technical Report NDN-0034, Revision 2.

[5] Junxiao Shi, Davide Pesavento, and Lotfi Benmohamed. 2020. NDN-DPDK: NDN Forwarding at 100 Gbps on Commodity Hardware. In Proceedings of the 7th ACM Conference on Information-Centric Networking (ICN '20). Association for Computing Machinery, New York, NY, USA, 30–40.

[6] NDN-DPDK [https://github.com/usnistgov/ndn-dpdk]

[7] NDN on FABRIC [https://github.com/initialguess/fabric-ndn/]

[8] NAC: Name-Based Access Control Library for NDN [https://github.com/named-data/name-based-access-control]

[9] Yingdi Yu. 2015. Name-Based Access Control NDN Tutorial. In Proc. of ACM ICN. https://named-data.net/wp-content/uploads/2015/11/ndn-tutorial-named-based-access.pdf

[10] Sharon Levy. Performance and Security of ECDSA. http://koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Levy.pdf

[11] FABRIC-NAC [https://github.com/agaouett/fabric_nac]