NDN Technical Memo: NDNS, NDN based Domain Name System

immediate

Revision history

Revision	Revision date	Description
0.21	Oct 22, 2014	draft, in the hope of comments/criticism/suggestion

1 Introduction

This report present NDNS (NDN-based Domain Name System) which solves the following operational chanenges of NDN network: undesirable namespace clashes or conflicts, cryptography support and scalable forwarding mechanism. NDNS achieves the goal by serving as an authoritative database that allows storing: 1) namespace delegation information to avoid namespace clashes or conflicts; 2) cryptographic information for content-verification; 3) mapping between routable and non-routable names. This report proposes a design for NDNS inspired by traditional DNS (Domain Name System) and DNSSEC (Domain Name System Security Extensions).

The ultimate goal of NDNS is to provide running, scalable, and universal data storage in NDN. The only precondition for using NDNS is that the data is named hierarchically compliant with NDN:

- Running means deployed. Thus we require a functioning system, not just a simple theoretical model.
- Scalable means it can support internet-scale service. The distributed and hierarchical structure used in DNS has proven to sufficiently support internet-scale service.
- Universal means supporting a variety of data types that allows the functionality of NDNS to be extended beyond a simple public key distribution service. For example, forwarding hints can be stored in NDNS to solve routing scalability and mobility issues.

This report is organized as follows: Section 2 gives the NDNS overview on naming, query, update and trust model. Then Section 3 presents the detailed design including database schema and packet format. The following Section 4 describe the management of NDNS. And Section 5 summarizes the report at last.

What is more, it is worth noting that the design in this report is an extension of the the one proposed in [7] with different message format, simplified mechanism and better security and efficiency.

2 Design

This section present the NDNS design. At the beginning of this section, the overview of NDNS mechanism, then NDNS naming convention, query, update and trust model is explained.

2.1 Overview

Figure 1 is the overview of NDNS. There are several concepts: namespace, zone, name server, caching resolver, and stub resolver. Basically, those concepts are inherited from traditional DNS [11].

The NDNS namespace is hierarchical and follows the namespace of traditional DNS. The domain namespace consists of a tree of domain names. Each node or leaf in the tree has zero or more resource records that hold information associated with the domain name. The tree is divided into zones beginning at the root zone. A DNS zone may consist one or more domains, The root zone is named with "/".

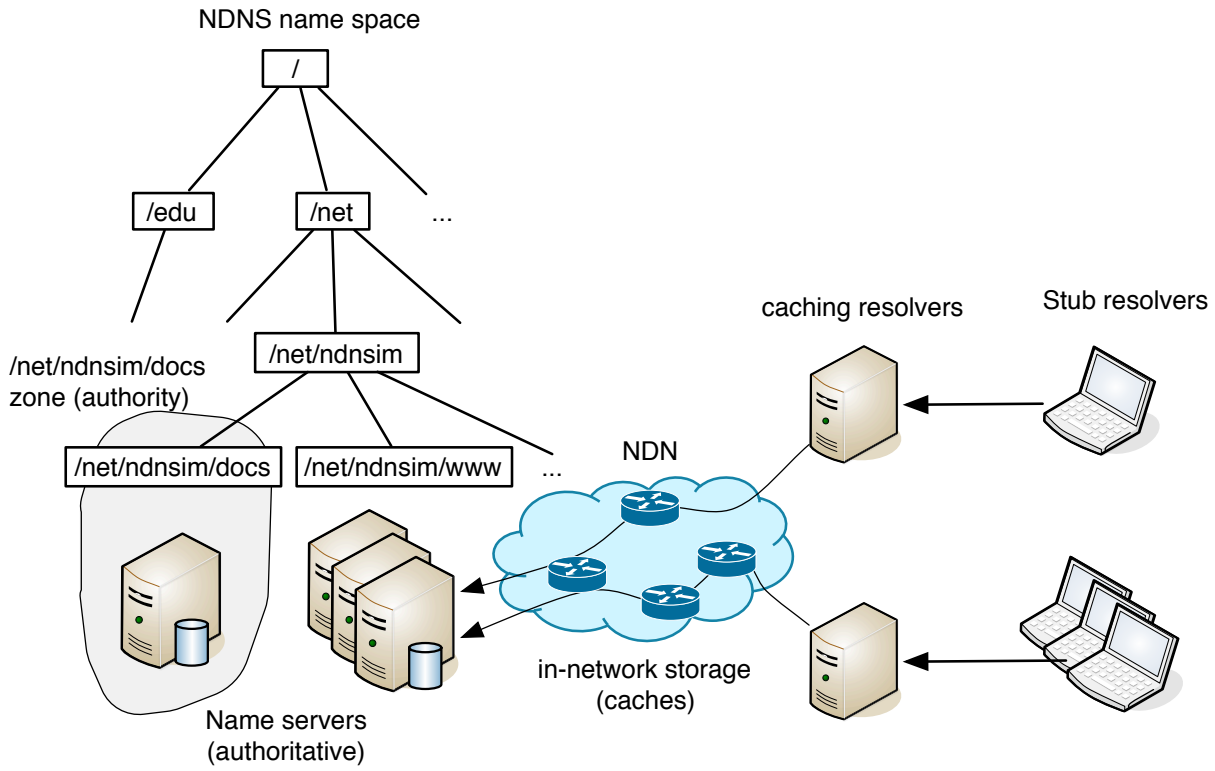

Figure 1: NDNS Overview

There are some significant differences between DNS/DNSSEC and NDNS.

- NDNS, allowing any Data stored at the system, is a universal data storage in NDN network, instead of mapping from one namespace to another namespace.
- In NDNS, the positive response (Data packet) of iterative query is constructed by authorized entity itself when record is created/updated, instead of name server when record is requested. In contrast in traditional DNS, the IP packets must be constructed on-the-fly, according to source and destination address. Responses in NDNS are static and independent of storage and destination locations.
- As caching is built into NDN, NDNS do not have to rely solely on caching resolver to cache all the results or intermediate responses. What's more important, NDNS is designed to ask questions as general as possible during iterative query, which could be shared with other queries. As a consequence, the importance of caching resolver is greatly reduced.

2.2 Naming Convention

Names in NDNS are just normal NDN Name described in [5]; however, the different components of NDNS name is application level design. In this section, we use two NDNS names, DSK (Data Signing) and KSK (Key Signing Key)[1] certificate names of Zone, e.g., /net/ndn/KEY/dsk-1234/ID-CERT/5678 and /net/KEY/ndn/ksk-1234/ID-CERT/5678:

The following table define name structure of NDNS:

[1] details will be explained later

Table 1: NDNS Name Convention

	Name Component	Data Type	KSK's Value	DSK's value
NDNS Name ::=	Authoritative Zone Name	NDN Name Type [5]	/net	/net/ndn
	Application Tag	BYTE+	KEY	KEY
	Label	NDN Name Type	/ndn/ksk-1234	/dsk-1234
	Type	BYTE+	ID-CERT	ID-CERT
	Version Number?	BYTE+	5678	5678

It is worth noting that Version Number is necessary for a full name, such as the name of specific resource record used for iterative query responses, furthermore response embed in Update message must contain a version number. But names of query and update messages and recursive query responses should not contain version numbers.

2.3 Query

There are two types of queries in NDNS, iterative and recursive. Iterative and recursive queries are described in traditional DNS [11]. NDNS inherit this methodology and concepts. Here we give a very short description of the types of queries.

Iterative query allows discovery of the existing mapping with minimal prior knowledge, i.e., the routable identifier (Name or IP address) of at least one root server. Name resolution begins at the root zone, immediate results are used to determine the next level of the query. This process continues until the name is fully resolved.

Recursive Query is used by stub resolver (most commonly implemented within the kernel providing a system call API). The traditional work-flow is a stub resolver sends a recursive query to caching resolver, who will adopt iteratively query to get the final result that is returned. The recursive strategy offloads the processing of DNS query to recursive resolver, which can offer benefits in many situations. For example, devices like sensors using recursive queries could save power and get result faster than using iterative query.

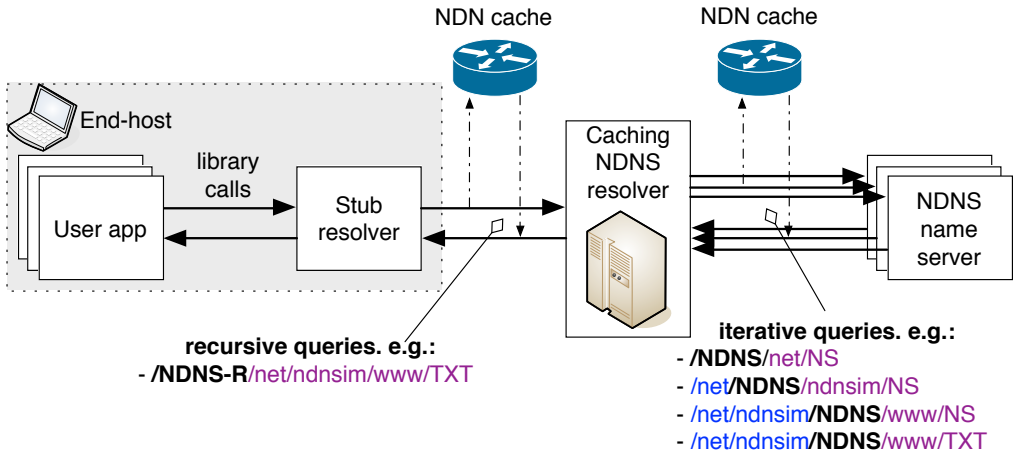

Figure 2: NDNS Query

Figure 2 shows the flow of a NDNS query. A stub resolver sends a recursive query to caching resolver, and caching resolver accepts the recursive query and translates it to a serial of iterative queries. Name server answers iterative query according to resource record (RR) stored at its database. Once caching resolver gets the final iterative query response, it constructs the recursive query response which embeds the final iterative

response inside. As we analyze above, the final positive response of an iterative query is constructed by the authorized application, thus, it is not necessary to sign the response of a recursive query with a time-consuming algorithm.

As shown in Figure 2, different questions are asked during every turn of iterative query. The questions range from general to specific step by step, and ultimate question is asked at last turn. While in DNS, the the ultimate question, is asked during every turn.

2.4 Update

When resource record changes, the corresponding entry in the NDNS system must be updated.

Update is initialized by the entity that owns the resource record. An authorized application sends an Interest, containing update message, towards NDNS authoritative server, who verifies the message and handles the update. Figure 3 shows the process that client follows to update the resource record.

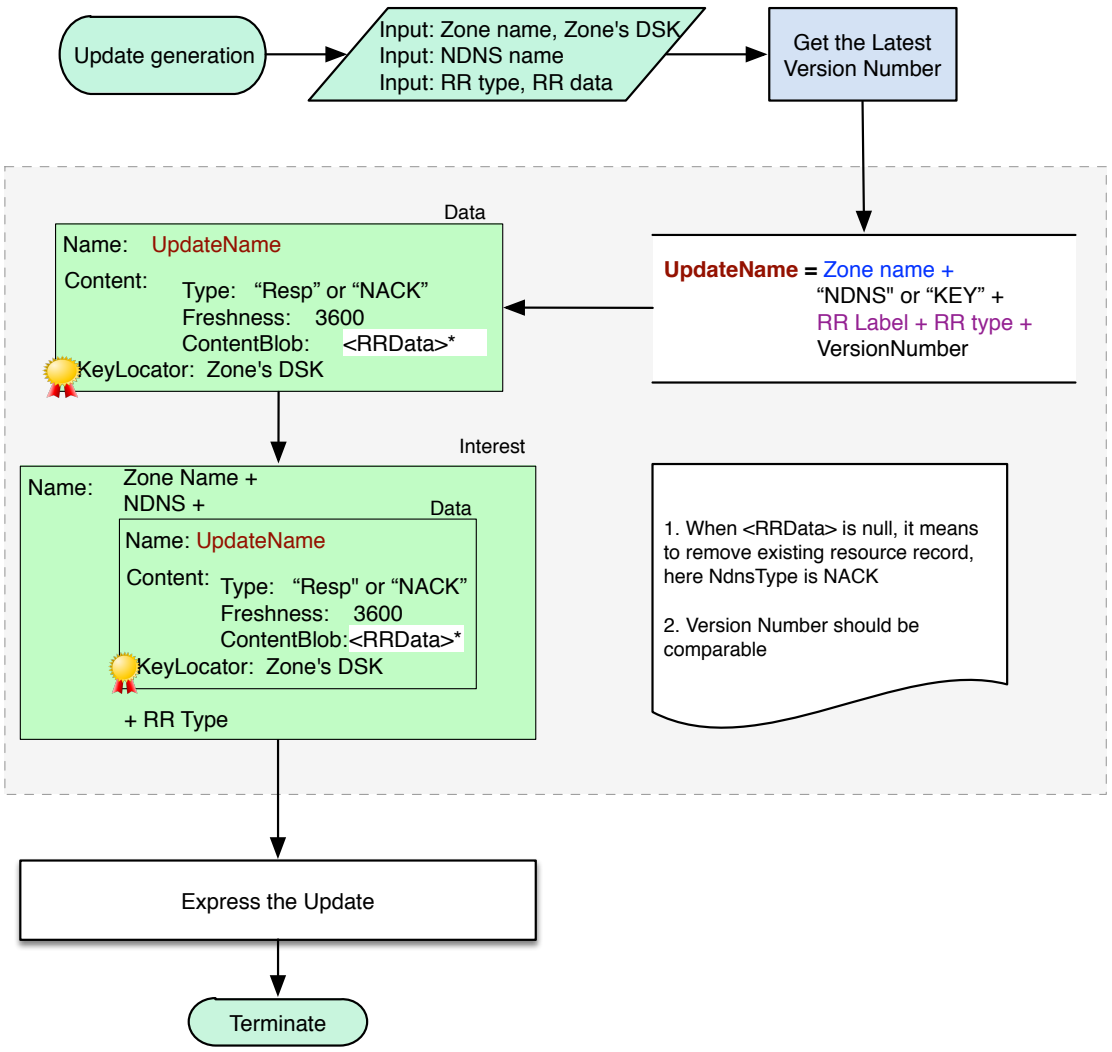

Figure 3: Client Update Process

There are multiple operations involved in completing an update, including adding/replacing/removing records. When an Update request embeds a response which does not contain any resource data, it intents

to remove an existing record.

Figure 4 presents the name server's process to update a record. When an authorized application requests to remove a record, authoritative name server do not "really" remove it, they instead substitutes existing one with the Nack response. The entry will not be removed until the key that signs the record expires. By doing this, NDNS defends itself from replay attack.

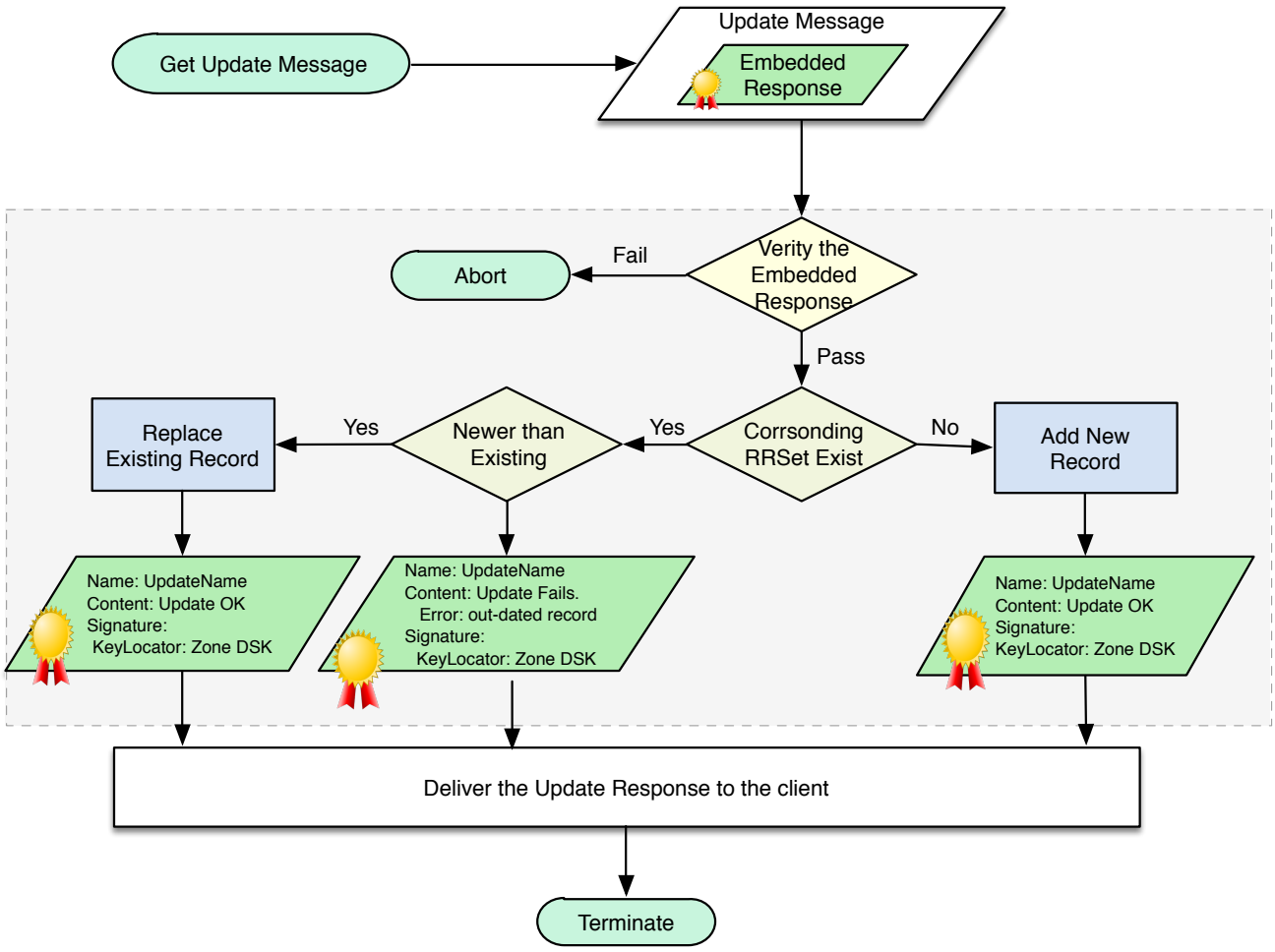

Figure 4: Name Server Update Process

Throughout the whole update process, security must be handled very carefully. The authorized application signs each update message, and authoritative zone only accepts update message from authorized application, verified through this signature, while malicious messages should be filtered. The response of an update request should be answered, and signed, by authoritative zone.

As it is quite common that one resource record is held by multiple name servers, database synchronization is required after update. This can be accomplished by using ChronoSync [12]. In NDNS, all the name servers logically behave as one. Thus in this report , we focus on NDNS level design and implementation, and ignore the synchronization, as is the case with current DNS.

When the update happens, there may be out-of-date cached responses in the NDN network and caching servers. All cached responses have a limited lifetime, therefore allowing simple strategies, such as setting a short lifetime, to limit side-effects of cached stale records. In this report we just ignore issues regarding stale

cached response.

2.5 Trust Model

As key distribution is one of the basic goal of NDNS, whose security, in turn, relies on key distribution itself, NDNS inherits many important concepts including: Zone Signing Key (ZSK), Key Signing Key (KSK), chain of trust, trust anchor from DNSSEC [8, 9, 10]. In NDNS, we substitute ZSK with Data Signing Key (DSK) given that key is used more general than just a zone in NDNS. The nationality for separation between ZSK and KSK is to allow zone owner to replace actual crytpo keys without contacting the parent authority/zone.

Data Signing Key (DSK) is used to sign Data packets. A packet's KeyLocator field is filled with the name of certificate of the DSK used to sign the key. While a KSK is used to certify each DSK. Each zone must own at least one valid KSK. In this way, a zone can certify many DSKs, each of which may be used by many identities under this zone. In Figure 5 we present an example.

The root zone is the trust anchor that provides a root key for the whole NDNS system. The root key should be able to be verified with out-of-band method. The root key is used as KSK of root zone. The root key certifies DSKs of the root zone by putting the root key's name to the KeyLocator field of DSK's certificate. Then the DSK certifies the KSK of zone "/net". Then DSK of zone "/net" is certified by its KSK. Then KSK and DSK of zone "/net/ndnsim" with the same method. Then every resource record delegated on zone "/net/ndnsim" could be certified by the DSK of this zone, or another key that could be verified be the KSK of zone "/net/ndnsim".

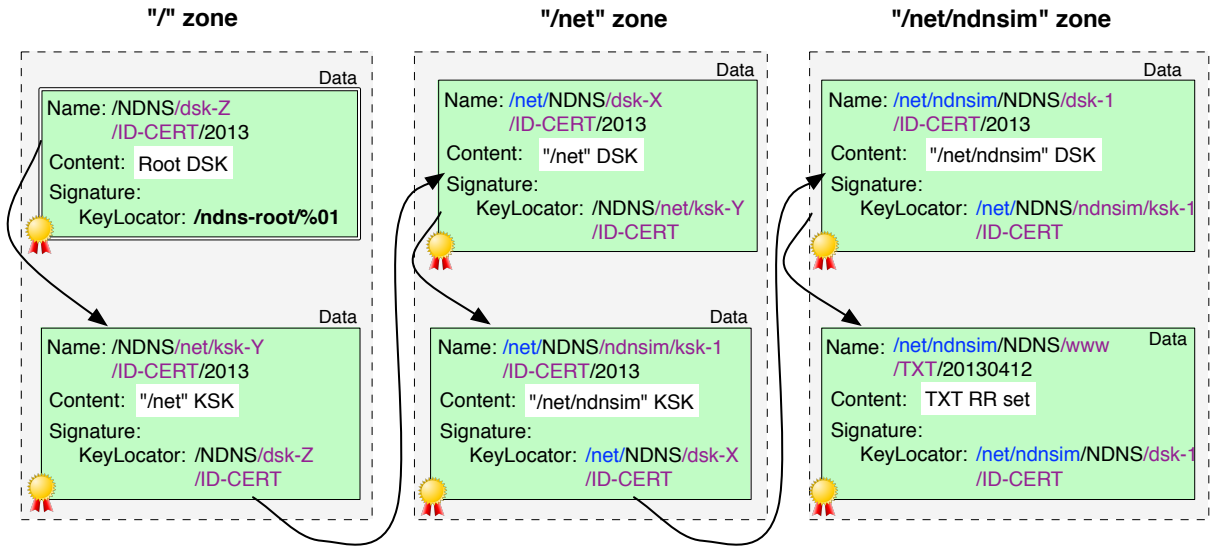

Figure 5: NDNS Delegation Example

3 Implementation

In this section, we explain the detailed implementation of NDNS, including database schema and format of NDNS packets.

3.1 Database

In NDNS implementation, two tables are created in database, zones and rrsets. Table zones are used to store NDNS zone's information, and Table rrsets is used to store resource record information. The detailed application related information is stored as wire format of iterative query response in Table rrset.

Table 2: Table Zone Scheme

Column	Data Type	Meaning	Constraint
id	uint64	identifier of each row	primary key
name	blob	zone's name	unique
ttl	unsigned int	Default TTL of the zone related rrset	cannot be null, 3600 is default value

Table 3: Table Rrset Scheme

Column	Data Type	Meaning	Constraint
id	uint64	identifier of each row	primary key
zone_id	uint64	foreign key to Table Zones	cannot be null
label	blob	label of name	cannot be null
type	blob	type of rrset, for example, ID-CERT	unique(zone_id, label, type)
version	blob	version of this record	can be null
ttl	unsigned int	TTL of this record	can be null, when the zone's ttl should be used
data	blob	the wired Data packet contains the record	can be null

In most cases, authoritative name servers do not construct Data packets, but rather store and respond to requests with well-formed pre-constructed Data packet directly. The pre-constructed Data could reused once the corresponding queries come. In this way, name servers may handle more queries and become more content-centric.

But after all, there must be some parties who construct the Data packet in the first place. In NDNS, this work is done during the delegation process.

3.2 NDNS Packet Format

In this section, we describe the format of NDNS packet, including: Query, Update and their Responses.

3.2.1 Query

Query packet is an Interest, which contains the NDNS Name. The format of Interest is described here [3], and the name convention is defined in Table 1. And the values of application tag is limited and shown in Table 4.

Table 4: Application Tag

Application Tag	Value	Comment
NDNS	Iterative query, except ID-CERT query	name server announce prefix ending with this tag
KEY	query for ID-CERT record	name server announce prefix ending with this tag
NDNS-R	Recursive query	caching resolver announce prefix ending this this tag

Note again, the name of query does not contain version number.

3.2.2 Iterative Query Response

NDNS allows universal Data packet stored in the system, which implies that it does not restrain the format of Response, the only assumption is that the Responses are NDN Data packet. However, in order to transmit NDNS-level data to support NDNS mechanism, a specific format of Content is defined.

Besides, NDNS defines an optional application meta information, named NdnsType in MetaInfo. Data packet is made up by multiple fields, including Name, MetaInfo, Content and Signature. The format is shown in Table 5.

Table 5: NDNS Response Packet Format

Response ::=	DATA-TLV TLV-LENGTH Name MetaInfo (=NdnsMetaInfo) Content Signature
Name	described in Section 2.2
NdnsMetaInfo	described in Section 3.2.2.1
Content	described in Section 3.2.2.2
Signature	described in Section 3.2.2.3

3.2.2.1 NdnsMetaInfo NdnsMetaInfo is derived type of MetaInfo [4]. Here we define the NdnsMetaInfo

Table 6: NdnsMetaInfo Format

NdnsMetaInfo ::=	NDNS-METAINFO-TLV(=METAINFO-TLV) TLV-LENGTH ContentType(=0, Blob) FreshnessPeriod NdnsType?
NdnsType ::=	NDNS-TYPE-TLV(=180) TLV-LENGTH nonNegativeIntegear

For an NDNS packet, the "ContentType" should always be "BLOB" (=0).
FreshnessPeriod is describe in [2].
NdnsType has some but limited values listed in Table 7.

Table 7: NdnsType Values

Value	Meaning
NDNS-RAW = 0	Raw application content is contained in the Response. Default value if NdnsType is not set.
NDNS-RESP = 1	Default value if not set. A positive NDNS Response. Positive means having the requested data
NDNS-NACK = 2	A negative NDNS response. Negative means cannot find the requested data
NDNS-AUTH = 3	Detailed question should be asked

Response contains a NdnsType in its MetaInfo field to distinguish packet type, which may not contain in normal Data packet. Those Response without containing NdnsType field explicitly is treated as positive response to NDNS query. The goal of NDNS-RAW is to allow universal data storing. For example, the certificate store at the name server, which does not contain NdnsType field, also serves as final response to a iterative query.

3.2.2.2 Content Its the end application which knows exactly the format of the content filed. So NDNS does not have more constraints on the standard NDN content format described here [1]. It is free to put raw application content in the content field. This kind of Response is NDNS-RAW.

However, as an application itself, NDNS also need a type of NDNS Data packet that transmits information among NDNS servers, such as name server information. Thus, NDNS define a specific content format that NDNS servers could understand listed in Table 8. When Response contains raw application content, it must be a NDNS-RAW Response.

Table 8: Content Format Used by NDNS Itself

Content ::=	CONTENT-TYPE TLV-LENGTH BYTE* | RR*
RR ::=	RR-TYPE-TLV(=190) TLV-LENGTH RRDATA
RRDATA ::=	RRDATA-TYPE-TLV(=191) TLV-LEGNGTH BYTE*

3.2.2.3 Signature Signature is described here [6]. In most cases, name servers do not construct and sign Data packet itself. There is one exception, Nack response is constructed by authoritative name server when none records (including Nack record) is stored at the name server.

3.2.3 Recursive Query Response

Recursive query response is normal Data packet, whose name is inherited from query Interest, and fill the content field with the wired format of final response of iterative query. Since embed response is already signed by authorized identity, it's not necessary to sign recursive query response. The format is shown in Table 9.

Table 9: Recursive Query Response

Response ::=	DATA-TLV TLV-LENGTH Name MetaInfo Content Signature
Name	Inherited from the Interest
MetaInfo	Described in [4]
Content	The final response of Iterative query
Signature	Described in [6]

3.2.4 Update

Update Message is a NDNS Request (Interest) which encapsulates a NDNS Response (Data) in its name. The format of name of NDN Query is described in Section 2.2. But in name of Update packet. The type of label component is not BYTE+, but a wired format of Response, whose NdnsType of the Response is NDNS-RESP. The name convention of Update message is defined in Table 11.

Table 10: Update Name Convention

	Name Component	Type of the Component
Update Name ::=	Zone Name	NDN Name Type [5]
	Application Tag (=NDNS)	BYTE+
	Embed Response	Wired Format Of Query Response, Section 3.2.2
	Type (=UPDATE)	BYTE+

3.2.5 Update Response

Update response is normal Data packet, whose name is inherited from Update Interest, and its content field is filled with update result message.

Table 11: Update Name Convention

Situation	Message in Content
Update Succeed	"Update OK"
Update Fails	"Update Fails. Error: "+append the error message

1. If update succeeds, the message should be: "Update OK".
2. if update fails, the message should be: "Update Fails. Error:" with error message appended.

Since Update is an Interest, the name server replies an Data. The Response to an update is also a Response which is signed by naming server.

4 Management

NDNS supports different types of resource records, specifically two are used in the management system. They are listed in Table 12.

Table 12: Common Resource Types

Type	Usage
NS	Name Server
ID-CERT	Certificate

NDNS management system needs NS and ID-CERT record types at minimal to make NDNS bootstrap.

Certificate in NDN is just Data packet. In NDNS, the certificate's KeyLocator is another name of certificate or root key/certificate. In current design, we assume that name server are reachable via zone name directly , it seems unnecessary to keep NS records since NDNS does not need another routable identifier by resolving NS. But NDNS still keeps NS for two reasons:

1. Maintain the hierarchical structure. NS record defines zone delegation relationship. Without it, caching resolver could not know when to stop iterative query.
2. For future use. It's possible that the name servers may need other routable identifier, such as forwarding hint. In that case, NS is a must.

In order to bootstrap NDNS, tools such as creating zone, delegating zone and generating certificate for DSK are essential. To be noticed, root zone is special, and therefore a specific tool is dictated to create root zone. What's more, tools like removing zone, undoing delegation, and removing root are also provided.

4.1 Create Zone

4.1.1 Input

- Zone Name

4.1.2 Output

- Local KeyChain: an identity named with zone name is added.
- Local KeyChain: a KSK is added.
- Local Filesystem: a KSK self-signed certificate with suggested file name pattern KeyName-SS.cert is added.
- Local NDNS Database: a new zone is added.

4.2 Delegate Zone

4.2.1 Input

- name of zone to be delegated
- its KSK self-signed certificate with suggested file name pattern KeyName-SS.cert
- authoritative zone name
- DSK name of authoritative zone; if not provided, the default DSK is used

4.2.2 Pre-Condition

- Authoritative zone must be an identity of Local Keychain system.
- Authoritative zone must be a prefix of the delegating zone.
- Delegating zone cannot be root zone.

4.2.3 Output

- Local Filesystem: delegating zone's KSK certificate issued by authoritative zone's DSK, with suggested file name pattern CertName-PKS.cert is added.
- Local NDNS Database: 1) an ID-CERT entry is added. The data is delegating zone's PKS cert. 2) an NS entry containing NDNS-RESP is added. 3) If the delegating zone has multiple layers of prefix, NS entries which contains NDNS-AUTH for every middle layer of suffix are added. For example, if a zone named /cs/alex is delegated at /, then an entry containing NDNS-AUTH for zone /cs will be added to the database.

4.3 DSK Certificate Generation

4.3.1 Input

- Zone name
- PKS certificate of zone KSK

4.3.2 Output

- Local KeyChain: a PKS certificate of KSK is added if keychain does not have it.
- Local File System: a DSK certificate signed by KSK with suggested file name pattern CertName-SKS.cert is aded.
- Local NDNS Database: an ID-CERT is added.

4.4 Root Zone

In NDNS system, Root Zone is special. Since it the root of the hierarchical system, Root Zone needs a special key. We call this special key as root key. Mechanism to verify this special key is out of the scope.

For Root Zone, it is unneccesary to create its KSK (and corresponding certificate). Still, a zone record should be added to the local database and its DSK should be registered to the local KeyChain.

4.4.1 Input

- The file path of the root key, or its certificate.

4.4.2 Output

- Local KeyChain: a DSK and corresponding certificate whose KeyLocator is the root key are added.
- Local Filesystem: a DSK certificate signed by root key with suggested file name pattern CertName-RKS.cert is added.
- Local NDNS system: an ID-CERT record is added.

4.5 Delete Zone

4.5.1 Input

- Zone Name

4.5.2 Output

- Local KeyChain: the identity named with zone name is removed.
- Local KeyChain: the KSK and its PKS certificate are removed.
- Local KeyChain: the DSK and its KSK-signed certificate are removed.
- Local Filesystem: KSK self-signed certificate is removed.
- Local Filesystem: KSK parent-signed certificate is removed.
- Local Filesystem: DSK KSK-signed certificate is removed.
- Local NDNS Database: the zone is removed.

4.6 Undelegate Zone

4.6.1 Input

- name of zone to be undelegated
- authoritative zone name
- path to the undelegated zone's PKS certificate

4.6.2 Pre-Condition

- authoritative zone must be an identity of local keychain system.
- undelegating zone should be delegated at the authoritative zone already.

4.6.3 Output

- Local Filesystem: undelegating zone's KSK certificate issued by authoritative zone's DSK, with suggested file name pattern CertName-PKS.cert is removed.
- Local NDNS Database: 1) an ID-CERT entry is removed. 2) an NS entry is removed. 3) If the delegating zone has multiple layers of prefix, NS entries which contains NDNS-AUTH for every middle layer of suffix are removed.

4.7 Delete Root Zone

4.7.1 Input

- The file path of the root key, or its certificate.

4.7.2 Output

- Local KeyChain: the DSK and corresponding certificate whose KeyLocator is the root key are removed.
- Local Filesystem: a DSK certificate signed by root key with suggested file name pattern CertName-RKS.cert is removed.
- Local NDNS system: an ID-CERT record is removed.

4.8 Further Discussion

Current management system design lacks a certificate transportation system. The reason for leaving this part blank is to provide flexibility for users to distribute the certificates. However, we notice that there is a need for a such built-in system, and therefore the design of an additional certificate management and transportation system is under discussed.

5 Summary

In this report , we introduce, NDNS, the domain name system for NDN. NDNS inherited some basic designs from DNS/DNSSEC, such as iterative query and recursive query, trust of chain, etc, but NDNS is designed for NDN, and makes fully use of NDN's architecture advantage, such as content-centric security, built-in network caching. In brief, we provide smooth security mechanism, simplified interactive process, and better performance.

Compared to previous work in [7], the fundamental ideas and concepts remain working, but this work improves its design and implements from the bottom, including the database schema, terminologies, behavior of name server, response verification, etc. The new solution provides better performance and becomes more content-centric.

References

[1] Ndn content type. http://named-data.net/doc/ndn-tlv/data.html#contenttype.

[2] Ndn freshness period. http://named-data.net/doc/ndn-tlv/data.html#freshnessperiod.

[3] Ndn interest format. http://named-data.net/doc/ndn-tlv/interest.html.

[4] Ndn metainfo. http://named-data.net/doc/ndn-tlv/data.html#metainfo.

[5] Ndn name format. http://named-data.net/doc/ndn-tlv/name.html#ndn-name-format.

[6] Ndn signature. http://named-data.net/doc/ndn-tlv/signature.html.

[7] Alexander Afanasyev. *Addressing Operational Challenges in Named Data Networking Through NDNS Distributed Database*. PhD thesis, University of California Los Angeles, 2013.

[8] Roy Arends, Rob Austein, M Larson, D Massey, and Scott Rose. Protocol modifications for the dns security extensions. Technical report, RFC 4035, March, 2005.

[9] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. Dns security introduction and requirements. Technical report, RFC 4033, March, 2005.

[10] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. Resource records for the dns security extensions. Technical report, RFC 4034, March, 2005.

[11] Paul V Mockapetris. Domain names-concepts and facilities. 1987.

[12] Zhenkai Zhu and Alexander Afanasyev. Let's chronosync: Decentralized dataset state synchronization in named data networking. In *Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP 2013)*, 2013.