NDSS: A Named Data Storage System

Shuo Chen, Junwei Cao*
Research Institute of Information Technology
Tsinghua National Laboratory for
Information Science and Technology
Tsinghua University, Beijing 100084, China
*jcao@tsinghua.edu.cn
Lipeng Zhu
Smart Grid Institute of State Grid Corporation of China
Nanjing 210003, China
zhulipeng@sgri.sgcc.com.cn

***Abstract*—NDSS (Named Data Storage System) is an architecture of distributed storage system which integrates local storage and networking with named data design. In traditional cloud and network storage system, local storage system and networking are separately designed. Storage data and network packet are encoded in different descriptions. NDSS is proposed to integrate the data description in both storage and network to reduce the overhead in data format transition and to provide new method for data operations. In this paper, the architecture and system design are proposed. Initial implementation design is also demonstrated based on Named Data Networking (NDN). The performance and functional comparison are conducted conceptually to show the advantages of NDSS.**

***Keywords*-Network Storage; Named Data Networking**

I. Introduction

The volume of network storage grows dramatically in this big data era. Many cloud and network storage solutions have been released, such as Google File System [1], Amazon S3[1], and Hadoop HDFS[2]. The architecture of large volume storage has transferred from Network Attached Storage and Storage Area Network architecture to large distributed file system working on commodity server and network device like GFS and HDFS. HDFS is reviewed for example. The basic storage unit of HDFS is data block. Data blocks are directly stored in file system on local machines. The meta data of data blocks are all managed by central nodes. Data block requests are transmitted through TCP/IP network, operated on local file system and also responsed through network. There is data description reinterpretation between network and local file system.

In [2], the layering architecture of application and network is reviewed. The performance of UNIX protocol stack and ISODE implementation of OSI upper layer application is evaluated. 97% of the total protocol stack overhead is attribute to the network data presentation function.

Named Data Storage System (NDSS) is proposed in this paper. The major contribution of NDSS is to integrate network and local storage data description with named data. Named data of network is not a new concept. Named data networking (NDN) is a clean-slate methodology compared to TCP/IP architecture. The network packets are not identified by the source or destination address, but hierarchically structured names in NDN. NDSS adopts named data to describe both network packet and local storage data and keeps the mapping between network or local storage location and uniform name of data. This is the conceptual approach of NDSS to integrate network and storage. An implementation design is also proposed based on NDN Forwarding Daemon (NFD) [3].

The rest of paper is organized as follows. Section II introduces the background of named data networking and storage. Section III demonstrates the architecture design of NDSS. Section IV introduces the implementation design of NDSS. Section V analyses the function and performance. Finally, section VI concludes the paper and addresses the future work.

II. Background

A. Named Data Networking

Named Data Networking (NDN) [4] [5] is a clean-slate data-oriented architecture different from TCP/IP. The key element of TCP/IP architecture success is the simple, universal network layer (IP) which realizes the global connections and supports most of the network functions. NDN also adopts this "thin" waist[3] design by replacing IP format packet with named data packet. The structure of NDN network stack is shown in Fig. 1.

NDN network packet includes *interest* packet and *data* packet. Name of NDN is hierarchically structured like URL. The *data* packet contains the name and content of the data. Besides this, *data* packet also contains the signature of to authenticate the data. *Interest* packet is the request of certain data and it contains the name of the requested data. The basic network transportation process is that the data requester sends the *interest* into NDN network and NDN network responds the *data* packet matching the name of the interest.

When an interest is sent by a consumer, it will be routed through NDN nodes. The architecture of NDN node is shown

[1]Amazon S3: http://aws.amazon.com/s3/
[2]HDFS: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[3]Thin waist of IP and NDN: http://named-data.net/project/execsummary/

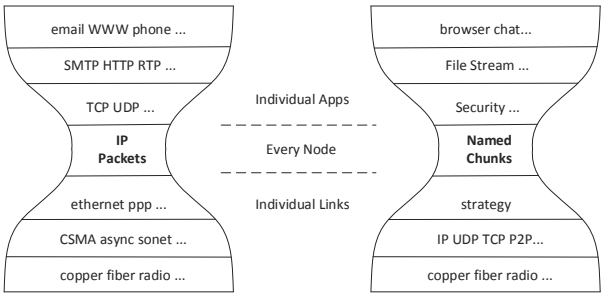

Fig. 1. IP vs NDN

in Fig 2. Forwarding Information Base (FIB) keeps records of name prefixes and faces to be sent, which means NDN node will look up the FIB to match the name of the interest and prefix of records to decide where to forward this interest. FIB is just like routing table in IP network. The difference is that IP routing table keeps prefixes of IP addresses and NDN FIB keeps prefixes of names. Pending Interest Table (PIT) keeps unsatisfied interests. The interest and its coming faces will be kept before matching data returns. Content Store (CS) is the cache of the data. When a data packet arrives NDN node, CS can keep this data for certain period for fast data acquisition.

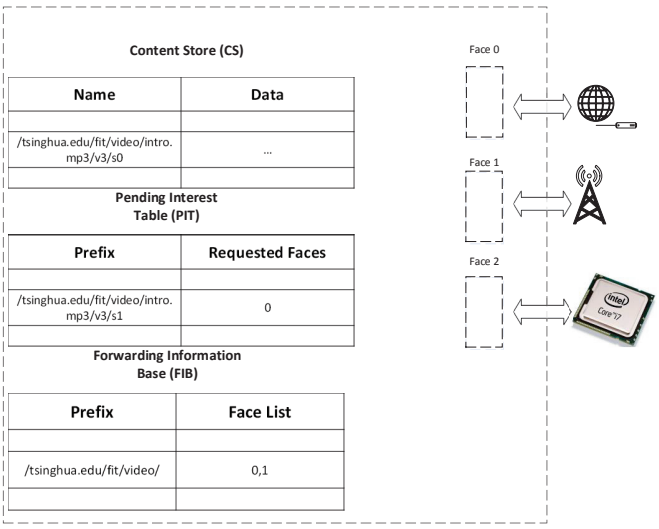

Fig. 2. NDN node

The typical process of NDN node handling interest packet is as follows:

a an interest packet is sent to an NDN node
b NDN node checks CS to look up whether there is a data packet matching the interest. If found, return data packet of the coming interest. If not, continue
c Put the interest into PIT and record its coming face.
d Look up FIB to match interest with longest-match prefix. If found, forwards the interest to the designated face. If not found, drop the interest.

B. Named Data Storage

Named Data Storage in this section is defined as certain type of network storage system, network packet and data block of which adopts the same format.

1) NDN Repository: NDN Repository (NDN Repo) is an persistent storage model to store named data over NDN network. It is an application to manipulate named data and to transfer data packet through NDN network. The operation of NDN repo includes reading from, insertion to and deleting from data objects in NDN repository nodes. A set of NDN repo protocol[4] is defined for semantics and operating process for persistent storage node in NDN. Once an interest is sent a NDN repo, the repo will search the local storage with the name of the interest. Once found, the repo will return the data packet matching the interest.

The repo protocol is a typical realization of application level framing [2]. NDN protocol just provides basic semantics and transportation process. Specific controls like congest control, flow control are all defined by upper repo protocol. In addition, the repo application directly handles the network named data.

Repo-ng (repo of new generation) is an implementation of NDN persistent in-network storage conforming to repo protocol. It uses ndn-cxx as NDN client library and database sqlite3 as underlying data storage. The named data packet is directly stored in the sqlite3 database and all the network interactions are through NFD.

The process of fetching data from repo-ng through NDN network stack is as follows:

a an interest is forwarded to the repo
b the repo query the named data database with the requested interest.
c if the named data matching the interest exists in the local repo. the repo will just responds the data packet through NFD.

2) Data-Centric Storage in Sensornets: Data-Centric Storage [6], [7] is proposed for sensornet which is a distributed sensing network. It demonstrates a primitive data distribution network with named data. In a large topology of network with 100,000 nodes, the Data-Centric Storage reduces the total network load and hotspot network usage as compared with external storage and local storage. This result shows the advantage of integrating network and storage with named data, in large data dissemination. The NDSS design will also adopt this advantage.

III. Design of Named Data Storage System Architecture

NDSS involves a major resign of distributed storage. In this section, we first illustrates the motivation and presents network and storage stacks design. The specific process is also demonstrated.

[4]Repo Protocol: http://redmine.named-data.net/projects/repo-ng/wiki

A. Motivation

The motivation of NDSS design is to reduce the redundant processes in transporting data between different hosts through network, due to the different description of network and storage data. For typical distributed file system like HDFS over TCP/IP network and posix file system, there are 2 major redundancies in data transportation process. One is the semantics conversion between network packet and application usable data. Once a network packet arrives the data node, it first converts extracts the content from the network packet and reorganizes these contents to data or command for application. The other redundancy is conversion between application readable data and data block on local storage drive. There is usually also a semantics conversion in this process. For HDFS, when the datanode fetches the data from the local filesystem, it should converts the high level location on data block to specific location on local file system.

For repo-ng, it reduces the first redundancy process that conversion between network packet and application usable data. The main reason is the naming semantics of NDN design. NDN adopts hierarchical structured naming like url. This naming convention can be directly used by the upper application.

Our NDSS adopts the NDN repo's application level framing design. Besides, NDSS makes several major improvements based on the architecture of NDN node to integrate the application level data and local storage data block. NDSS also proposes a more flat network stack based on NDN. The specific design is demonstrated on the following sections.

B. Named Data Design

The key approach to integrate network and storage is by named data. Compared with NDN design, *interest* and *data* packets are introduced to serve as network request and response. The motivation of NDSS semantics design is to propose uniform data request and response for both network and local storage data. For common application dealing with file system, the application fetches certain piece of data by identifying the file name and location of the file. This mechanism is naturally suitable with NDN naming mechanism. However, for file system like linux or windows, posix api is conventional standard and system call is the uniform interface, which is practically different from how NFD works.

In NDSS design, the request and response of data is designed in abstract level. NDSS also defines *interest* and *data* for data request and response. *Interest* carries the name of data requested and selectors. Selector[5] in NDN is defined as extra constraints to select the data. NDSS proposes an extra selector named *priority*. Since network and local storage are two sources of data, the major function of *priority* is to choose the prior data source.

In NDN, signature is a must section of data packet. While if an application fetches the self published data from local storage, there is no need to authenticate the data. In NDSS, tag *local* is proposed to guarantee the local source. The constraints of tag *local* is that this data is published by local host and never transported outside the host. If an outside data packet arrives the local node or an inner data packet is forwarded to the outer network, the node would first converts the *local* tag to false.

C. NDSS Node Model

NDSS data transportation is driven by the consumers of data. NDSS borrows data transmission mechanism from NDN that Data is transmitted only in response to an *interest* and consumes that *interest*. [4] In NDN, interests are forwarded to network according to FIB. While in NDSS, the interest is not only forwarded to network, but also handled by underlying local storage. The FIB is modified as Local and Forwarding Information Base (LFIB). The architecture of NDSS node is shown as Fig. 3:

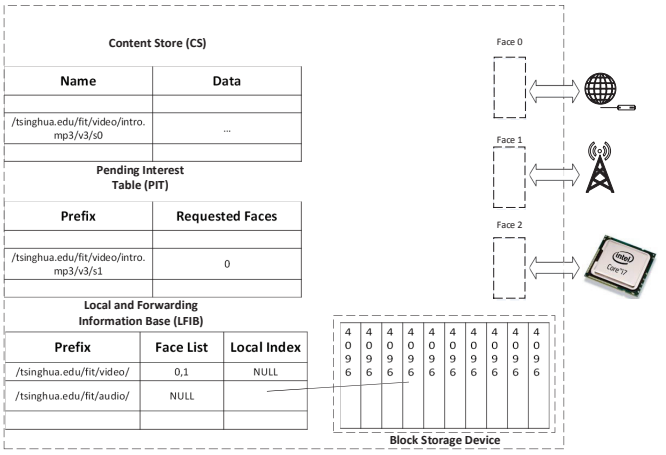

Fig. 3. NDSS node

As shown in Fig. 3, NDSS adopts CS and PIT table as NDN node. CS serves as the cache of the data and PIT preserves the outgoing and unsatisfied requests. The key approach to integrate network and local data is redesign of the FIB to LFIB. Compared to NDN FIB, LFIB includes an extra column local index. For one entry in LFIB, it not only records the faces to request the data from network, but also the specific index on local storage device. Block device such as hard disk is the mainstream settings for current hosts. The local index directly locates the specific block on the block storage device by recording the location. Thus, when an interest arrives at the node, NDSS node can query the data by the same metadata table.

D. Flat Network Stack

Network protocol of NDSS adopts most of the NDN process. The common implementation of NDN such as CCNx[6] or NDNLP [8] is deployed upon traditional address routing network, usually upon TCP/IP network. These overlay designs are easy to implement and could adopts advantages of

[5]http://named-data.net/doc/ndn-tlv/interest.html#selectors

[6]http://www.ccnx.org/what-is-ccn/

current network architecture. However, this layered network stack produces multiple overheads besides NDN. For example, CCNx, named data packets is split and reassembled through TCP/IP stack. Besides, the routing process should be executed twice in both NDN router and common IP router. Most network issues such as flow control, congestion control and reliable communication could be handled within NDN network architecture. In [2], Application level framing is proposed to achieve Integrated Layer Processing for efficient data manipulation. The NDSS network design adopts Application Data Unit (ADU) to reduce the overheads in data transmission. Compared with overlay network, the NDSS design is named as underlay network. The architecture of NDSS network design is show as Fig. 4.

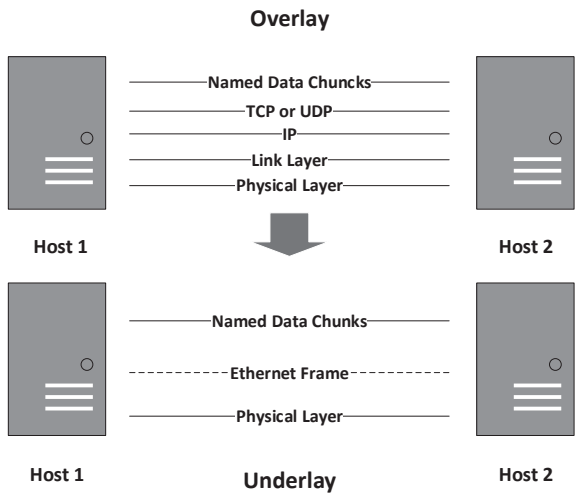

Fig. 4. NDSS Underlay Network

Compared with overlay design upon TCP/IP network, the named data protocol directly runs on link layer Ethernet in underlay design. For NDN network, the only constrain for underlying network is to guarantee hop by hop transmission. In NDSS design, the MAC address of Ethernet header is removed.

E. NDSS Data Transportation Process

The motivation of NDSS design is to reduce overhead of data transmission. The NDSS node model is to reduce the overhead from network to storage device. The NDSS underlay network is to reduce between hosts through network. Processes of data transportation are introduced in this section.

1) Local Application Request Local Data:

a Application sends request for certain data
b Check if CS holds any data matching the request. If found, return the data and quit the process
c Check if PIT holds the same interest. If found, put local application identifier into the PIT and quit the process
d Check LFIB table. If there is an entry matching the interest, there are 2 conditions to select data from local storage: interest sets priority selector as local or local index is selected by forwarding strategy.
e The data block is directly selected by the local data index.
f Data is responded according to the PIT entry and the matching entry is removed.
g The named data block is inserted into the CS.
h Application gets the responded data packet.
i Check the local tag of the data block. If the local tag is false, application should validate the signature of the named data block according to application defined security policy.

2) Local Application Request External Data:

a Application sends interest for certain data
b Check if CS holds any data matching the interest. If found, return the data and quit the process
c Check if PIT holds the same interest. If found, put local application identifier into the PIT and quit the process
d Check LFIB table. If there is an entry matching the interest, forward the interests according to the face list.
e Wait for the responded data packet.
f Extract the data block directly from the ethernet frame. This data can be directly used by the upper application.
g Remove the matching entry in PIT.
h The named data block is inserted into the CS.
i Validate the signature of the data packet.
j If upper application would store the data packet, the data packet can be directly assigned in the block of storage device. The local index of such name would be updated in the LFIB.

3) NDSS Node Handles the External Interest:

a NDSS node receives interests from external faces.
b Check if CS holds any data matching the interest. If found, return the data and quit the process
c Check if PIT holds the same interest. If found, put local application identifier into the PIT and quit the process
d Check LFIB table. If there is an entry matching the interest, node will forward the interest according to the face list. If local index is also the target, the named data block is directly selected by the local data index.
e The local tag of data block is set to be false.
f Data is responded according to the PIT entry and the matching entry is removed.
g The named data black can be directly encoded in the ethernet frame and be sent.
h The named data block is inserted into the CS.

IV. Design of NDSS Implementation

The implementation of NDSS consists of two major parts: NDSS model and redesign of network stack. The network stack is implemented and evaluation is demonstrated in next section. The implementation design of NDSS model is proposed in this paper and the practice will be introduced in future works.

A. NDSS Model Implementation Design

In conventional system, upper application fetches local data through local file system and network data through sockets. In

linux, these data sources can all be accessed by file descriptor through *read()* and *write()* methods. In NDSS design, the unit of data access is based on block and the application can directly accesses data block instead of file. If application wants to execute data upon file level, file system module could be built upon NDSS like NDNFS [9].

The initial implementation of NDSS node is based on Linux shown as Fig. 5. A block device is mounted on the VFS and provides basic *read()* and *write()* methods. Raw socket is used for directly encapsulating named data block into the Ethernet Frame.

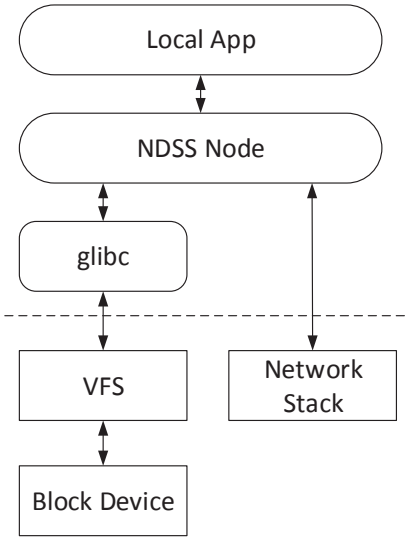

Fig. 5. NDSS Node Implementation

B. Network Stack Implementation

Ethernet is used as NDSS underlying protocol. There are two key points to be handled: size of named block and removal of ethernet header.

Ethernet does not provide segmentation and the common MTU is 1500. The common named block size is more than this. The size of Ethernet frame could be handled by Jumbo frame. Jumbo frames are Ethernet frames with more than 1500 bytes of MTU. Conventionally, jumbo frames can carry up to 7000 bytes of MTU. Most Gigabit Ethernet NICs support jumbo frames.

The ethernet frame without MAC address will be dropped, since NIC will filtered ethernet frames by destination address. Promiscuous . In our NDSS implementation, the network interface is set as promiscuous mode.

V. Evaluation of NDSS Design

A. Evaluation of NDSS Process

Compared with the conventional layered system, NDSS reduces the following processes:

a conversion between TCP/IP packet and named network data.
b redundant forwarding according to IP FIB.
c conversion between network content between application usable data
d conversion between network usable data and local storage data.

B. Evaludation of NDSS Underlay Network

Underlay network of one hop between 2 hosts is implemented. Data throughput is tested using CCNx project *ccngetfile* tool. The result is shown as Fig. 6. There is a slight performance drop between underlay and TCP. The major reason is that TCP/IP protocol stack is well-tuning in the kernel, while implementation of raw socket is not optimized.

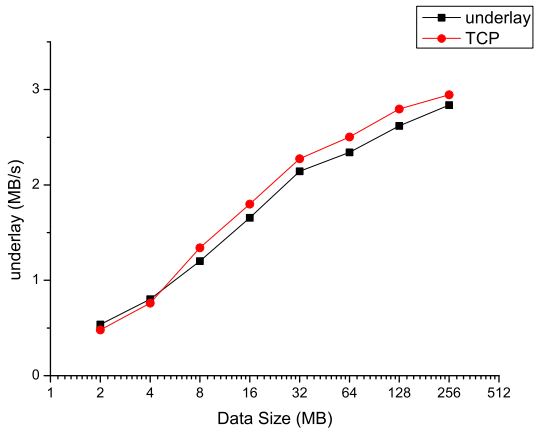

Fig. 6. One Hop Throughput

VI. Conclusion

NDSS idea is inspired by the concept of application level framing. The principle of NDSS is to deeply reduce the overhead in data transmission between hard disks through network. The design of LFIB is the key to integrate network and strorage by named data. The process of data operation is discussed and network performance is evaluated. There is a reasonable performance drop due to the unoptimized implementation. There are many works in future such as integrating the NDSS into the kernel, building access and privacy control over NDSS, and implementing metadata system over NDSS.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China (grants No. 61472200 and No. 61233016), Ministry of Science and Technology of China under National 973 Basic Research Program (grant No. 2013CB228206), and S&T projects of State Grid Corporation of China.

References

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[2] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *ACM SIGCOMM Computer Communication Review*, vol. 20, no. 4. ACM, 1990, pp. 200–208.

[3] A. Afanasyev, J. Burke, P. Crowley, S. DiBenedetto, J. Thompson, B. Zhang, and L. Zhang, "An introduction to ndn and its software architecture," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 1–2.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

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

[6] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: a geographic hash table for data-centric storage," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. ACM, 2002, pp. 78–87.

[7] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 137–142, 2003.

[8] J. Shi and B. Zhang, "Ndnlp: A link protocol for ndn," *The University of Arizona, Tucson, AZ, NDN Technical Report NDN-0006*, 2012.

[9] W. Shang, Z. Wen, Q. Ding, A. Afanasyev, and L. Zhang, "Ndnfs: An ndn-friendly file system."