Underlay Implementation of Named Data Networking

Junwei Cao[1,2*], Shuo Chen[1,3], Zhen Chen[1], Ge Ma[1,3], Zhongda Yuan[1], Ziwei Hu[4], Jing Zhou[4], and Jinghong Guo[4]

[1]Research Institute of Information Technology, Tsinghua University, Beijing, 100084, China
[2]Tsinghua National Lab for Information Science and Technology, Beijing, 100084, China
[3]Department of Automation, Tsinghua University, Beijing, 100084, China
[4]State Grid Smart Grid Research Institute, Beijing 102209, P. R. China
Corresponding email: jcao@tsinghua.edu.cn

Abstract. Named Data Networking (NDN) is a paradigm shift from a traditional host-to-host network to a named content based one by a wholly new designed network architecture. CCNx, which is the most popular implementation of NDN, is designed over TCP or UDP. While the current implementation of NDN is an overlay network, the major contribution of this paper is to propose an underlay implementation of NDN by using Ethernet frames directly to encapsulate named data even without a MAC address. We present an underlay NDN prototype. In this paper, pure named content architecture is proposed, with a concrete implementation and performance evaluation using Ethernet frames. Our underlay implementation has a slightly better theoretical protocol efficiency than an overlay implementation. The experiment itself also demonstrates the feasibility of pure NDN, which is an essential evolution from a traditional address based to a content based network.

Keywords: Named Data Networking, Content-Centric Networking, Underlay Implementation

1 Introduction

The original motivation for networking is to share data or expensive computing resources with host-to-host communication in the mainframe era. Today's Internet is based on the ossifying TCP/IP protocol stack and a static host-to-host conversation model to disseminate contents with ever larger volumes. Named Data Networking (NDN), also called Content Centric Networking (CCN), is a new architecture recently proposed by PARC [1][2]. The fundamental principles of NDN have been provided by some designs such as Content-Oriented Architecture or Information-Centric Networking (ICN). Some related designs are DONA [3], TRIAD [4], PSIRP [5] and so on. NDN is a new network architecture that redesigns the model of network communication, from today's focus on *where* - addresses and hosts, to *what* - the content that users and applications care about. Because of this key feature, basic network functions

such as routing, forwarding and security are named data based instead of link state or session based.

The common implementations of NDN, such as CCNx [6] or NDNLP [7] are deployed upon the traditional address routing network. The overlay design is easy to be implemented, adapts to existing Internet architecture and could utilize some mature strength of the current network architecture including reliable communication, abundant routing optimization algorithms and so on. However, this overlay design deepens the protocol stack and produces unnecessary overhead. For example in CCNx, NDN packets are split and reassembled through the TCP/IP stack. Besides, the routing process should be executed twice in both the NDN router and the common IP router. In [3], main stream network architectures are reviewed. Application level framing is proposed to achieve Integrated Layer Processing for efficient data manipulation. We propose an implementation of NDN over Ethernet frames and call it underlay NDN implementation. MAC addresses are removed from the headers of Ethernet frames and the link is just hop-by-hop. The principle of underlay NDN is that the NDN protocol directly runs on the layer 2 link protocol without host addresses, since the original motivation of NDN is to shift from a *where* to a *what* model.

In our work, the CCNx code is modified to support Ethernet frames without host addresses. Each network interface card (NIC) which handles the transmission of underlay NDN packets is abstracted as a NDN *face*. The NDN packet will be transmitted through the face using a non-address frame protocol. The layer 2 framing protocol provides an agreement between the two ends of a physical link, which is the basic demand to support NDN. Figure 1 shows the framework of our design, which is a little different from the NDN overlay protocol stack. In addition, an overall design of an underlay NDN network is proposed and a testbed of a mini network is deployed to prove the functionality and evaluate the performance.

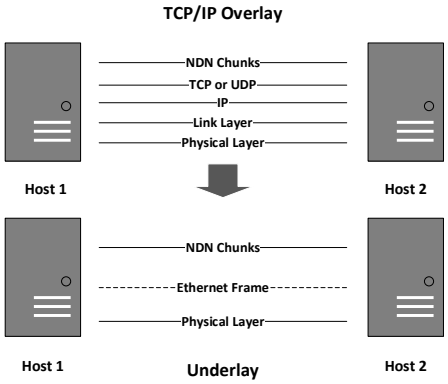


Fig. 1. The NDN underlay model

For performance evaluation, *tcpdump*[1] is used to analyze the structure of underlay NDN packets for protocol efficiency. The CCNx repository (CCNR) is used as the data source for NDN clients to fetch. Different topologies of the undelay NDN net-

[1] Tcpdump: http://www.tcpdump.org/

work are deployed for the CCNR data distribution to test network delay and throughput. Besides, we use the VLC application over underlay NDN to demonstrate the function integrity.

The rest of the paper is organized as follows. Section 2 gives some research background and related work. Section 3 describes detailed implementation of underlay NDN and the overall design of underlay NDN. Section 4 and 5 demonstrate testbed deployment, and performance evaluation of different scenarios with detailed discussions. Section 6 concludes the paper and addresses the future work.

2 Research Background

2.1 CCNx

CCNx [6] is an open source project and a software prototype that implements the NDN architecture. NDN has no notion of hosts at its lowest level, and the functionality of NDN is guaranteed by the name based routing property. The specification of the CCNx protocol implements the notion of NDN. It prescribes the format of interests and content objects, the naming and encoding specification of NDN packets, CCNx node models and so on. The CCNx node model implements three main data structures of NDN: Content Store (CS), Forwarding Information Base (FIB) and Pending Interest Table (PIT). The CCNx node runs as a daemon *ccnd* to process the NDN protocol, and takes charge of forwarding interests referring to the FIB table, caching NDN packets in CS and responds to interests with content objects according to PIT.

The current implementation of CCNx is an overlay upon TCP or UDP. The configuration of FIB is done by the program *ccndc*. It can bound NDN faces with TCP or UDP socket file descriptions. The transmission of interests and content objects is through NDN faces. In CCNx, the formatted NDN packets are sent through TCP and UDP sockets.

The NDN application development framework is shown in Figure 2. The configuration command program of CCNx, such as *ccndc*, and CCNx applications bounded with the CCNx project, such as *ccnr* are all based on the CCNx client library. The CCNx client is a local socket communication program. The command or application messages are encapsulated as CCNx interests or content objects and sent by local CCNx clients with local socket communication to *ccnd*. The CCNx client will also register NDN faces in *ccnd*. The *ccnd* daemon is the only communication portal among different CCNx hosts. If a developer wants to develop a new application of CCNx, he or she is just required to call the CCNx client library to connect with the *ccnd* daemon and follow the CCNx naming and interest specifications. Thus the modification of the *ccnd* exterior communication mechanism will not influence CCNx commands and applications.

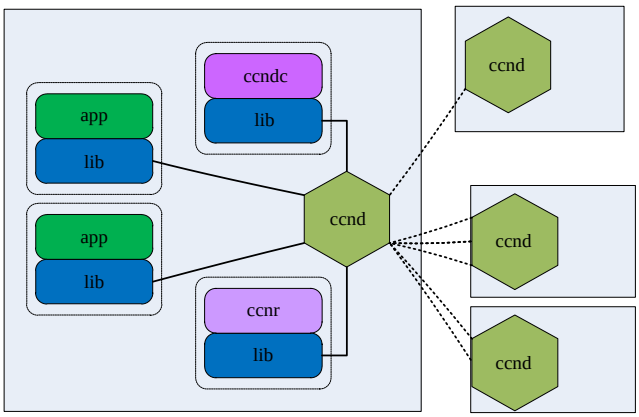


Fig. 2. The NDN application development framework

2.2 Related Work

In [8], the layered network architecture is revisited. The modular design will result in multiple presentation conversions between different layers which contributes to majority of the protocol stack overhead. Application level framing is proposed to promote this presentation process to the application.

The first paper which proposes the NDN architecture [1] evaluates the performance of the NDN implementation. The result shows that TCP throughput asymptotes to 90% of the link bandwidth and overlay NDN asymptotes to about 68% of the link bandwidth. The bulk data transfer efficiency of overlay NDN is comparable to TCP but lower due to its large header overhead. This paper also compares the relative performance of overlay NDN with HTTP and HTTPS to retrieve a single HTML file. In this experiment, two different NDN encapsulations are compared: directly into 1500 bytes Ethernet packets which have no IP or UDP headers and into UDP datagrams directly. The overhead of the two performances seems very similar to that of HTTP. In the performance evaluation of underlay NDN, we will refer to these two experiments to analyze NDN throughput and content efficiency.

Haowei Yuan et al. [9] evaluated the content distribution performance of overlay NDN based and HTTP based content distribution solutions using the CCNx code. The performance evaluation shows that the current overlay NDN prototype implementation is much slower than TCP. This paper also demonstrates that more than 50% of the time is spent on functions related to packet name decoding in the overlay NDN prototype. The *ccn_skeleton_decode* function, which is the lowest level packet decoding function, takes 47.34% of the all the packet processing time [10].

Junxiao Shi et al. proposed a link protocol for NDN, NDNLP [7]. NDNLP runs between the CCN chunks and underlying network protocols including TCP, UDP and Ethernet links. The NDNLP protocol receives NDN packets and sends it to lower links. NDNLP provides two main features: fragmentation and reassembly to support different sizes of packets, acknowledgement and retransmission to support reliable

transmission. The current implementation is the mode of proxy. The daemon *ndnld* receives CCNx packets from *ccnd*, encapsulates and sends out them in the format of NDNLP, receives remote NDNLP packets, decodes them and sends them to local *ccnd*.

Our previous efforts on overlay networks include performance comparison between content delivery networking (CDN) and NDN [11], node placement of overlay networks for Internet of Things applications [12], building services [13] and data storage [14] upon NDN, and bitmap indexing for big data applications [15]. This paper mainly discusses the underlay implementation.

3 Underlay Implementation

In this section we will present the implementation of underlay NDN.

Figure 3 shows a basic deployment of underlay NDN. In our implementation, the Ethernet frame protocol is modified to be the NDN underlying layer 2 protocol. The MAC address is removed from the Ethernet header. NDN faces are bounded with NIC and different faces are directly connected by physical links. We implement this work by revising the CCNx code and building the testbed on Linux servers. In the following section, we will introduce enabling techniques firstly and briefly demonstrate the modification of CCNx codes and configurations on Linux servers.

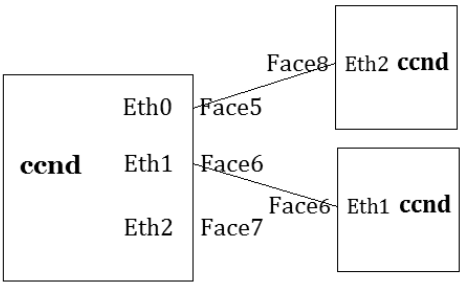


Fig. 3. The NDN underlay deployment

Because of the removal of MAC addresses, the underlay NDN packets cannot go through common Ethernet switches. In addition, NDN nodes cannot connect by common IP routers because of lack of IP headers. Thus, the underlay NDN network of our implementation just consists of CCNx nodes bounded with underlay faces. Thus, the pure underlay NDN network should be specially designed. One of the demonstration systems is described in the following section.

3.1 Enabling Techniques

1) *Jumbo frames.*

Although the default maximum transmission unit (MTU) of Ethernet frames is 1500 bytes in most operation systems, the MTU can be changed. As mentioned above, the

MTU is changed to support the large size of NDN segments in our work. Many NICs support larger Ethernet frames, which are called jumbo frames[2]. 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.

In the protocol of CCNx, the default size of segments is 4096 bytes. Thereby every segment should be fragmented due to the limited 1500 bytes MTU. We change the MTU to 7000 bytes so that the NDN segment has not to be fragmented. Furthermore, we will also use different sizes of segments to make full use of the payload.

2) *Promiscuous mode.*

Promiscuous mode[3] is a mode for a wired network interface card or wireless network interface card that will not filter all traffic it receives rather than just receive the Ethernet frames which matches pending MAC addresses. In our implementation of underlay NDN, there is no MAC headers at all (its space is occupied by the payload). Therefore, we should use promiscuous mode to pass all frames that are received. Most of the NICs support the promiscuous mode.

3) *AF_PACKET and Raw socket.*

AF_PACKET is the first parameter of function *socket(int socket_family, int socket_type, int protocol)*. This AF_PACKET sockets are used to receive or send raw packets at the device driver (OSI Layer 2). The *socket_type* could be SOCK_RAW for raw sockets[4] or SOCK_DGRAM for cooked packets with the link level header removed. In our implementation, we use raw sockets since headers of Ethernet frames could be handled directly. Thus, MAC addresses could be removed from Ethernet headers.

3.2 Implementation

In this section, we will demonstrate how to build a one-hop underlay NDN prototype with two Linux servers over Ethernet frames. The CCNx code is enhanced to support Ethernet frame socket transmission and underlay NDN face configuration. Linux servers are configured to support our non-address Ethernet frames.

1) *CCNx code modification.*

Unlike the current NDNLP implementation, we directly modified the CCNx code. The major work can be summarized into two parts. The first part is to revise *ccnd* related code to support raw socket communication. The second part is to revise *ccndc* related code to add underlay NDN face supports. The brief process of modification can be shown in the following figures.

---

[2] Jumbo frame: http://en.wikipedia.org/wiki/Jumbo_frame

[3] Promiscuous mode: http://en.wikipedia.org/wiki/Promiscuous_mode

[4] Raw socket: http://en.wikipedia.org/wiki/Raw_socket

Figure 4 shows the face registration process and figure 5 shows the message receiving process. We just add some flags or macros to judge the type of protocol or type of faces. The practical modification is a case branch and the mount of modification is rather small.

2) *Network configuration.*

We choose CentOS5.5 as the experiment platform. Our server's CPU is Intel(R) Pentium(R) Dual CPU E2160 @ 1.80GHz and memory is 4G. NIC is TP-LINK TG-3269C. The configuration of the network is listed below:

- To support jumbo frames of larger MTU, for example 7000, the Linux command may be as follows:

  sudo ip link set dev eth0 mtu 7000

  The parameter *eth0* is the name of the bounded NIC. The command including the following configurations requires root authority.

- The underlay NIC should be detached from TCP/IP protocol stack, in case that the NIC would receive TCP/IP packets that the underlay NCN faces could not handle.

  sudo ip addr flush dev eth0

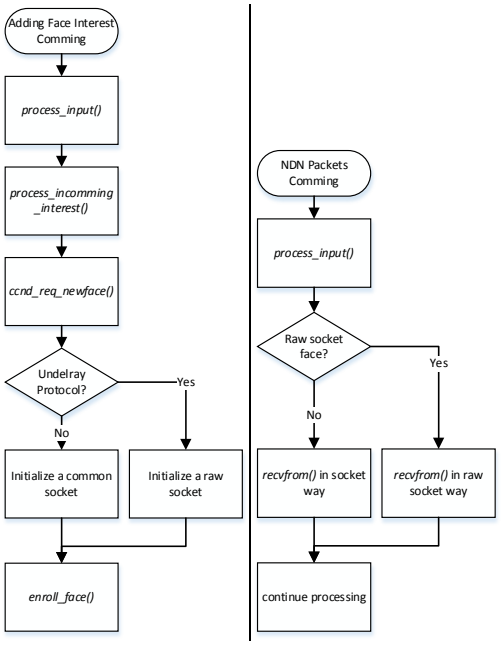


Fig. 4. Face registration proces **Fig. 5.** Message receiving process

4 Performance Evaluation

4.1 Building of Testbed

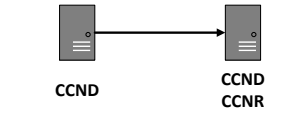
The building of the testbed could build on the one-hop demonstration deployment. Testbeds of different topologies are deployed for different evaluation scenarios. In this paper, one-hop, two-hop and star topologies are demonstrated as show in Figure 6.

One-hop deployment as shown in Figure 6(a) is demonstrated before. The server on one end runs the CCNx repository (CCNR).

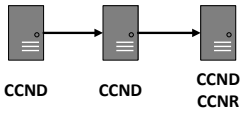
The two-hop deployment as shown in Figure 6(b) is rather similar. The middle node initiates two underlay NDN faces and runs two underlay specific NICs. The two ends connect to the middle node's two NICs. One end of server runs CCNR.

The starred topology as shown in Figure 6(c) is also similar. The central *ccnd* node has three NICs connected with border *ccnd*.

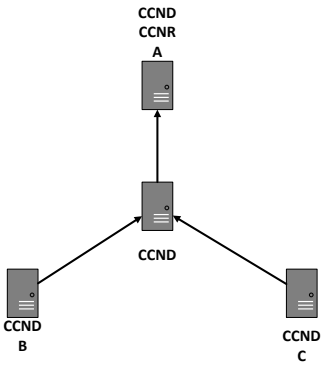
The direction of *ccndc* configuration is the direction of the arrows in the figure, which means the direction of forwarding.



(a) one-hop



(b) two-hop



(c) starred

Fig. 6. Network topologies

4.2 Bulk Data Transmission

In this section, we compare the NDN underlay implementation with NDN over TCP on network throughput. The CCNx repository *ccnr* is used to store the bulk data and the command *ccngetfile* is used to fetch files from *ccnr*. In particular, for the same size of the data, we first fetch the file from A to B, and then from A to C. Because of the cache of the central *ccnd*, we can expect better throughput from the latter scenario.

Figures 7, 8 and 9 shows network throughput comparison between NDN underlay with NDN over TCP with different sizes of bulk data on one-hop, two-hop and starred topologies. Beyond our expectation, we can see that the throughput of underlay is a littler lower than that of overlay implementation.

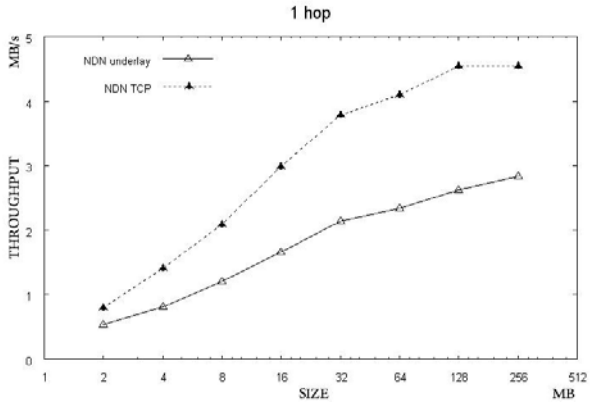


Fig. 7. One hop performance

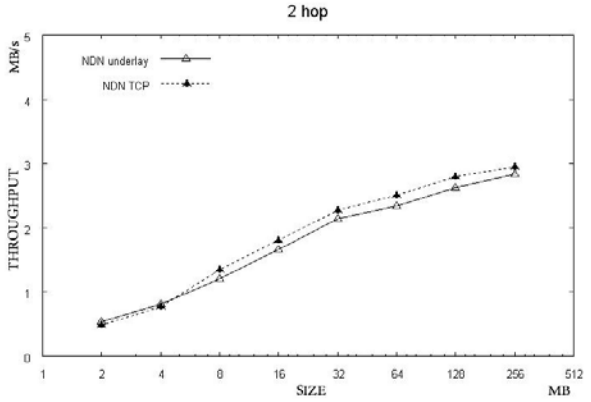


Fig. 8. Two hop performance

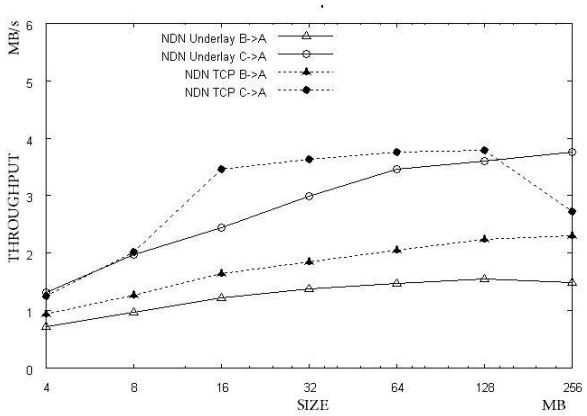


Fig. 9. Starred topology performance

One of the expected properties of underlay NDN is higher throughput than overlay NDN. We will discuss this result in Section V-A.

4.3 Protocol Efficiency

In the prior experiment, we found that for the default block size 4096, the full size of one NDN packet is 4772. For TCP with MTU of 1500 bytes, the whole NDN packets will be split into 4 TCP packets with 54 bytes of TCP and IP headers. The protocol efficiency is defined as the ratio of the practical size of message with the total size of message. As shown in Table 1, the protocol efficiency of NDN underlay a little higher than NDN over TCP.

Table 1. Jumbo frame structure of MTU 7000

Network Type	Protocol Efficiency	Calculation
NDN TCP	82%	4096/(4772+54*4)
NDN Underlay	86%	4096/4772

4.4 Video Streaming Demo

The VLC plugin which is in the CCNx code can support video streaming applications [16]. The installation and execution process of VLC in CCNx-underlay is the same as that of CCNx. The code of the VLC plugin has not been modified from the original CCNx code.

The network topology of VLC is two hop as shown in Figure 6(b). One VLC client requests streaming videos from the other *ccnr* end through the middle of *ccnd*. In our experiment as shown in Table 2, the input bitrate of the video is 468kb/s and the content bitrate is 489kb/s, which means normal quality for watching.

Table 2. Result of VLC over underlay NDN

Video	
Decoded	383 blocks
Displayed	351 frames
Lost	1 frames
Input/Read	
Media data size	2974 KiB
Input Bitrate	468kb/s
Content Bitrate	489kb/s

5 Discussion

5.1 Performance Analysis

In Section IV-B, we can see that network throughput of the underlay NDN is lower than that of the overlay NDN, which is beyond our expectation. Theoretically, the underlay NDN has simpler underlying protocols and should have better performance. For CCNx over TCP sockets, CCNx packets are split and reassembled through the TCP/IP stack. In addition, because of the IP and NDN double routing protocol, the same NDN packets will be forwarded duplicately. In Section IV-C, the NDN undelay has lower protocol overhead than NDN overlay. Thus, the NDN Underlay should have a better performance theoretically.

We speculate that the code efficiency of raw sockets may be the main bottleneck and check the raw socket related code of the Linux kernel as follows. The kernel version is 2.6.36.2. As we can see in the function *__raw_v4_lookup*, it will check the type of message protocol and filter the address. This is one of the codes which will degrade the performance and there may be other un-optimized codes. We think that un-optimized code is one of the main reasons for low performance. The highly optimized TCP/IP layer works so fast with the Ethernet layer, that the underlay NDN implementation talking to the Ethernet layer directly did slow things down.

We will improve the CCNx underlay code by revising the Linux kernel code to optimize raw sockets or just injecting a new transmission tunnel over the NIC driver in future work.

In conclusion, we demonstrate that with NDN, TCP/IP is no longer needed, but also demonstrate that if TCP/IP is omitted, the replacement needs to be efficiently embedded, thus making optimization of surrounding layers and the OS-kernel necessary.

5.2 Architecture Analysis

As demonstrated above, our implementation of underlay NDN network is the pure content based network.

Reliable hop-by-hop links are the only demand for the NDN protocol. The underlay NDN flattens the protocol stack and relaxes the layer 2 protocol requirements, which enhances the underlying link layer compatibility. The data

representation will directly be possessed to NDN level to avoid presentation overhead. Data manipulation and transport control could be configured and more targeted to achieve selective features.

In addition, the complete removal of the TCP/IP stack simplifies the control and data plates of the NDN overlay network. As we can see in papers that propose NDN [1][2], the NDN architecture itself is a complete and functional network architecture. The essential principle of the underlay NDN is to leave the routing, forwarding, security and other network concerned problem with the NDN architecture itself.

6 Conclusions

In today's Internet, high efficiency of data transmission is still very difficult and important. In this paper, we present a new NDN protocol implementation. We design the underlay NDN model and implement our design. It is an evolution from traditional address-related infrastructure dependencies. Although the current implementation of underlay NDN, CCNx-underlay, cannot surpass the TCP or UDP overlay on network throughput, we believed that performance of CCNx-underlay will be better than the overlay implementation with optimization of network programming.

In future, we will still develop and optimize the CCNx-underlay project and release new versions. We will also enhance CCNx-underlay to support more underlying protocols. In addition, we think the naming mechanism of NDN can integrate networking, storage and computing and we will propose a new architecture of distributed computing in future work. More applications on top of our underlay NDN implementation will be developed in future, e.g. in the areas of Internet of Things, Smart Grid and Energy Internet.

7 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 State Grid R&D project "Research on the Architecture of Information Communication System for Internet of Energy" (grant No. SGRIXTKJ[2015]253).

8 References

1. Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L.: Networking named content. In: Proceedings of the 5th international conference on Emerging networking experiments and technologies, pp. 1-12. Rome, Italy (2009)
2. Zhang, L., Estrin, D., Burke, J., Jacobson, V., Thornton, J. D., Smetters, D. K., et al.: Named data networking (ndn) project. Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC (2010)

3. Koponen, T., Chawla, M., Chun, B. G., Ermolinskiy, A., Kim, K. H., Shenker, S., et al.: A data-oriented (and beyond) network architecture. ACM SIGCOMM Computer Communication Review. Vol. 37, pp. 181-192 (2007)
4. Cheriton, D. R. and Gritter, M.: TRIAD: A new next-generation Internet architecture, http://www-dsg.stanford.edu/triad/triad.ps.gz, 2000
5. Lagutin, D., Visala, K., and Tarkoma, S.: Publish/Subscribe for Internet: PSIRP Perspective. Future Internet Assembly. Vol. 4, pp.75-84 (2010)
6. The Project CCNx, http://www.ccnx.org/.
7. Shi, J. X. and Zhang, B. C.: NDNLP: A Link Protocol for NDN. NDN Technical Report NDN-0006 (2012)
8. Clark, D. D. and Tennenhouse, D. L.: Architectural considerations for a new generation of protocols. ACM SIGCOMM Computer Communication Review. Vol. 20, No. 4, pp. 200-208 (1990)
9. Yuan, H. and Crowley, P.: Experimental evaluation of content distribution with NDN and HTTP. In: The 32nd IEEE International Conference on Computer Communications, pp. 240-244. Turin, Italy , (2013)
10. Yuan, H., Song, T., and Crowley, P.: Scalable ndn forwarding: Concepts, issues and principles. In: Computer Communications and Networks (ICCCN), 2012 21st International Conference on, pp. 1-9. Munich, Germany, (2012)
11. Ma, G., Chen, Z., Cao, J., Guo, Z., Jiang, Y., and Guo, X.: A Tentative Comparison on CDN and NDN. In: Proc. 2014 IEEE Int. Conf. on Systems, Man and Cybernetics, pp. 2893-2898. San Diego, CA, USA, (2014)
12. Wan, Y., Cao, J., He, K., Zhang, H., Yu, P., Yao, S., and Li, K.: Node Placement Analysis for Overlay Networks in IoT Applications. Int. J. Distributed Sensor Networks. Vol.1, pp. 103-110 (2014)
13. Chen, S., Cao, J., and Zhu, L.: Named Service Networking. In: Proc. 10th IEEE Int. Conf. on Networking, Architecture, and Storage, pp. 315-320. Boston, MA, USA, (2015)
14. Chen, S., Cao, J., and Zhu, L.: NDSS: a Named Data Storage System. In: Proc. 2015 IEEE Int. Conf. on Cloud and Autonomic Computing, pp.196-199. Cambridge, MA, USA, (2015)
15. Wu, Y., Chen, Z., Wen, Y., Cao, J., Zheng, W., and Ma, G.: A General Analytical Model for Spatial and Temporal Performance of Bitmap Index Compression Algorithms in Big Data. In: Proc. 24th Int. Conf. on Computer Communications and Networks, pp. 1-10. Las Vegas, NV, USA, (2015)
16. Xu, H., Chen, Z., Chen, R., and Cao, J.: Live streaming with content centric networking. In: Networking and Distributed Computing (ICNDC), 2012 Third International Conference on, pp. 1-5. Hangzhou, China, (2012)