Name Label Switching Paradigm for Named Data Networking

Jiangtao Luo, *Member, IEEE*, Chao Wu, Yi Jiang, and Jingwen Tong

***Abstract*—Named Data Networking (NDN) is regarded as one of the promising architectures of future Internet, in which every packet has a name and packet forwarding is based on lookup of name other than IP address. Scalable fast packet forwarding is always a challenge in NDN. In this letter, a MPLS-like label switching mechanism is proposed, called Name Label Switching (NLS), which uses fixed-length label swapping replacing unbounded name lookup at core nodes and caches data only at edge nodes. The NLS node architecture and forwarding process were presented, and its forwarding performance was analyzed and evaluated by simulation. It showed that NLS can improve the forwarding performance significantly. Using NLS, about 37% of the data response time and about 70% of the Interest packets processing time at core nodes can be saved, compared to native NDN.**

***Index Terms*—Label switching, named data networking, forwarding.**

I. Introduction

SINCE 1970s, the network architecture based on the TCP/IP protocols is faced with many increasingly serious challenges, such as network security, reliability, flexibility, mobility, congestion control and resource allocation [1]. As we know, the existing network implements end-to-end data transmission between hosts through switching data packets, of which host-based addressing is the cornerstone. However, as the applications of content distribution explosively develop and grow to prevail, the inefficiency of the existing architecture becomes more notable. Therefore, many variants of the Information-Centric Networking (ICN) are worked out to fulfill the need of the future network.

Named Data Networking (NDN), a typical representative of ICN, is regarded as one of the promising architectures of future Internet, in which every packet has a name and packet forwarding is based on lookup of name other than IP address. Communication in NDN is driven by the receiving end, i.e., the data consumer. To receive data, a consumer sends out an Interest packet, which carries a name that identifies the desired data [2]. When an Interest arrives at a node, the node first checks the local content cache, called Content Store (CS), for the name. If the requested data has been cached before, a Data packet will be sent back together with its name to the consumer; otherwise, a new entry will be appended to the Pending Interest Table (PIT), which contains the arrival interfaces of Interests that have been forwarded but still waiting for matching data. Then, the Interest will be forwarded by looking up its name in the Forwarding Information Base (FIB), which is populated by a name-based routing protocol [3].

Thus, name lookup is a key issue to be addressed in NDN which forwards packets by examining their hierarchical and variable length names instead of IP addresses [2], [3]. Especially, it is always a big challenge to implement scalable fast name lookup at line rates in designing NDN routers. Recently, many methods have been explored to optimize the efficiency of name lookup [4]–[6]. Varvello [4] presented a design scheme of high end router that supports name-based forwarding using Bloom filter. A scalable Name Lookup engine with Adaptive Prefix Bloom filter (NLAPB) was reported [5], in which each NDN name/prefix is split into B-prefix followed by T-suffix. B-prefix is matched by Bloom filters whereas T-suffix is processed by the small-scale trie. A two-stage Bloom filter based scheme for name lookup was reported in [6], in which the first stage determines the length of a name prefix, and the second stage looks up the prefix in a narrowed group of Bloom filters based on the results from the first stage. In their work, forwarding throughput was all reported to be significantly improved and memory consumption was reduced.

Nevertheless, the end-to-end forwarding performance cannot be expected to be improved too much just by lookup algorithm optimization without trying to reduce the lookup times. In fact, at each hop, the name of an Interest packet will probably be examined more than once: twice for exact matching at CS and PIT, once for the longest prefix matching at FIB, unless the interest is fortunately matched at the first query.

Enlightened by the mechanism of Multi-Protocol Label Switching (MPLS), we want to investigate the benefit from adopting label switching mechanism in NDN forwarding by creating a MPLS-like core to accelerate the looking up process and reduce the total times of name lookup. In an MPLS network, data packets are assigned labels according to its destination at entrance nodes of MPLS, called Label Edge Router (LER), then packet-forwarding decisions are made solely on the contents of this label, without the need to examine the packet itself. Before being assigned labels, packets are partitioned into different Forwarding Equivalence Classes (FECs), which are mapped to different Label Switched Paths (LSPs) throughout the MPLS core [7]. In this letter, we would like to verify the idea and evaluate its performance.

II. Architecture

A. Concepts

Consider a network model illustrated in Fig. 1. A client or consumer sends Interest packets to an NDN access network.

Manuscript received August 10, 2014; accepted December 19, 2014. Date of publication January 1, 2015; date of current version March 6, 2015. This work was supported in part by the Chongqing Municipal Application and Development Planning Project under GRANT cstc2013yykfA40006 and Program for Innovation Team Building at Institutions of Higher Education in Chongqing under GRANT KJTD201312. The associate editor coordinating the review of this paper and approving it for publication was G. Y. Lazarou.

The authors are with the Electronic Information and Networking Research Institute, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: luojt@cqupt.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2014.2387344

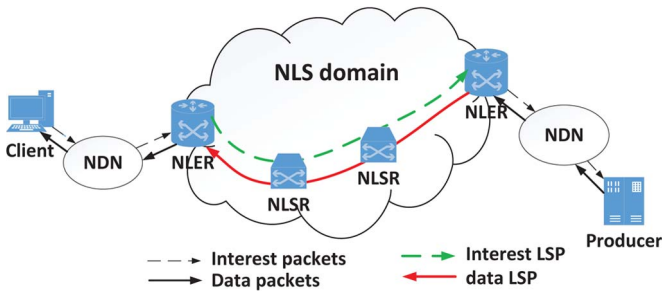

Fig. 1. Name Label Switching domain.

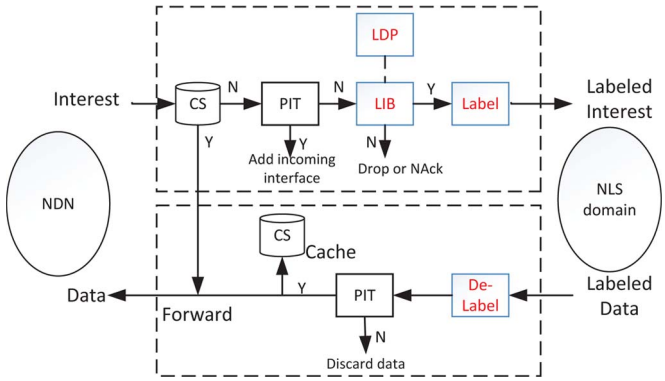

Fig. 2. Forwarding process at an ingress node.

The data provider or producer is located at another NDN network. They are connected by the proposed label switching core, denoted by *Name Label Switching* (NLS) domain analogous to MPLS domain, which forwards packets based on label swapping replacing native name lookup.

This architecture has the following features:

- NLS domain connects different NDN client networks and provides Interest and Data delivery service for them.
- NLS domain consists of two kinds of nodes: edge nodes and core nodes. An edge node, also called *Name Label Edge Router* (NLER), puts a label on each incoming Interest or Data packet according to its FEC, or erases its name labels when it leaves the NLS domain. A core node, generally called *Name Label Switching Router* (NLSR), just forwards packets based on label swapping.
- The labeled Interest and Data packets are delivered on predefined *Name Label Switching Path* (NLSP), built by Label Distributed Protocol (LDP). There are two kinds of NLSP, Interest LSP (ILSP) an Data LSP (DLSP), transferring Interest packets and Data packets, respectively. Their difference lies mainly in the setup procedure.
- The matching data is cached only at edge nodes, required by label swapping at core nodes no need for CS any more.
- An edge node may be an egress or ingress one, dependent on the direction of Interest packets, i.e., the ingress node receives Interest packets entering NLS domain while the egress one sends Interest packets out of NLS domain.

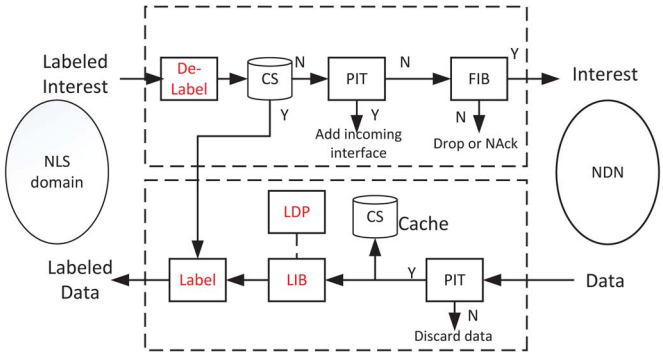

Fig. 3. Forwarding process at an egress node.

B. Node Architecture and Packet Forwarding

The architecture of NLS edge nodes and packet forwarding processes are illustrated in Figs. 2 and 3. The pseudo-codes of forwarding procedures for Interest and Data packets are also shown in Algorithms 1 and 2, respectively.

Algorithm 1 Interest packet forwarding in ingress nodes.

Input: *interest*, the Interest packet arriving at the ingress node.
Precondition: ILSP and DLSP have been built.
Main program:

```
1: if interest matched found in Content Store then
2:     return matching DATA
3: else
4:     if a matching entry found in the PIT then
5:         add its coming interface to the matching PIT entry.
6:     else
7:         if a matching entry found in the LIB with its ILSP
   then
8:             label the interest; send the labeled interest to the
   next hop.
9:         end if
10:    end if
11: end if
```

As shown in Fig. 2, when an Interest packet arrives at an ingress NLER from an NDN client network, the node first checks the CS for matching data; if it exists, the data will be sent back on the interface where the Interest comes. Otherwise, the node will examine the PIT, and if there is a matching entry, the incoming interface of the Interest will be appended to the interface list in the entry. Otherwise, the node will search for a proper label in the Label Information Base (LIB) using name longest prefix matching, which contains entries consisting of name prefix, input/output interface and labels populated by LDP. If matched, the Interest will be labeled and forwarded into the NLS domain along predefined ILSP. On the opposite direction, when a labeled Data packet arrives from the NLS core, the node first erases the label and then handles it in the same way as an NDN router [3]. Additionally, the matching Data will be forwarded further into NDN client and a duplicate of it will be saved in the local cache (CS).

Algorithm 2 Data packet forwarding in egress nodes.

Input: *data*, the Data packet arriving at the egress node.
Precondition: ILSP and DLSP have been built.
Main program:
1: **if** *data* name not found in the PIT **then**
2: **drop** *data*.
3: **else**
4: Cache the *data* in local CS.
5: Search label in the LIB using its name.
6: Label the *data*; send the labeled *data* to the next hop.
7: **end if**

Forwarding process at an egress node is shown in Fig. 3. When a labeled Interest arrives from the NLS domain, the node first erases its label and checks the CS for matching data; if it exists, the data will be labeled and sent back on the predefined DLSP into the NLS domain. Otherwise, the Interest will be processed as in a native NDN node. On the opposite direction, when a Data packet arrives from NDN client, the node first checks the PIT for unsatisfying Interests; if found, the node will search the LIB for a proper label to label the Data and forward it on the DLSP. And one duplicate is saved in the CS for future Interests.

The architecture of NLS core nodes is relatively simple. It just completes label swapping and forwarding according to LIB as in MPLS domain, without checking packet names and data caching, thus accelerating packet forwarding in NLS core.

C. Setup of NLSP

ILSP and DLSP have different setup mechanisms. ILSP will be created due to the change of content distribution, e.g., when a new piece of data is saved in the CS at an edge node or when the edge node receives the latest content broadcast from connected NDN nodes. To keep the forwarding nature of NDN, DLSP is built symmetrically to the corresponding ILSP on the opposite direction (Fig. 1). Whenever an entry for a new ILSP is created in the LIB, a new entry will also be created immediately for its DLSP just by exchanging the input/output interface and corresponding labels.

As an example, Table I demonstrates the LIB entries in the NLER after an ILSP and its DLSP were created for a name prefix "/cq/mpg." Note that the last field (Ord.) in the entry is a virtual one just showing the order in which the entry was created. The example showed that after the ILSP for "/cq/mpg," $\{(5,200)\} \rightarrow \{(3,200),(7,100)\} \rightarrow \{(4,100)\}$ was created, the related DLSP was also built on the opposite direction, $\{(4,100)\} \rightarrow \{(7,100),(3,200)\} \rightarrow \{(5,200)\}$.

TABLE I
LIB AT EDGE NODES AFTER NLSP SETUP

Node	LSP	Name	Inf_i	L_i	Inf_o	L_o	Ord.
Ingress	I	/cq/mpg	-	-	5	200	3
-	D	/cq/mpg	5	200	-	-	4
Core	I	-	3	200	7	100	2
-	D	-	7	100	3	200	5
Egress	I	/cq/mpg	4	100	-	-	1
-	D	/cq/mpg	-	-	4	100	6

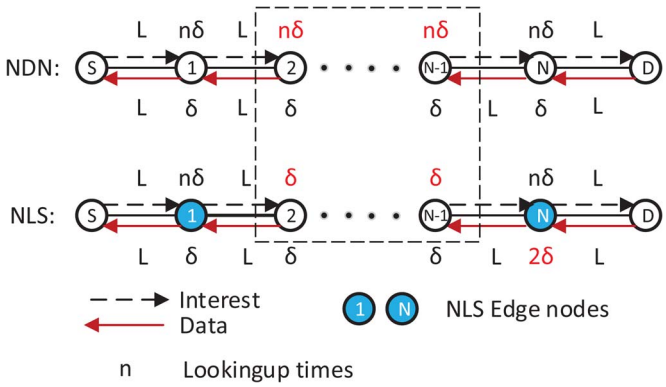

Fig. 4. Analysis of response time in native NDN and NLS.

III. PERFORMANCE EVALUATION

In this section, forwarding performance of NLS was evaluated. Data response time, denoted by RT, was taken as the metric, which is defined as the time between sending an Interest and receiving the corresponding Data.

Consider a linear topology in Fig. 4 and the RT can be analyzed as follows. For simplicity, assume that the time required for once name lookup and label searching is identical, denoted by δ, although in general, label searching is much faster than name lookup. Besides, assume L, the propagation delay of each link segment; N, the number of hop counts; n, average times of name lookup in each NDN router, possibly 1, 2 or 3. Then, the response time using NDN can be evaluated by $RT_{ndn} = 2(N+1)\cdot L + N\cdot(n+1)\cdot\delta$, whereas the response time using NLS can be evaluated by $RT_{nls} = 2(N+1)\cdot L + (2N+2n-1)\cdot\delta$. The major difference happens at NLS core nodes and egress nodes. At NLS egress, an extra δ is required to search label for matching data. At core nodes in NLS, only δ is enough to process a labeled Interest while $n\cdot\delta$ is needed to process an Interest at an ordinary NDN node. Accordingly, the end-to-end data response time saving by using NLS can be evaluated by $\Delta = RT_{ndn} - RT_{nls} = [(n-1)\cdot N - (2n-1)]\cdot\delta$. Therefore, when n is larger than 1 (almost always true), and N is above 3, the response time saving will be obtained, proportion to the number of hop counts.

Simulation results showed the same tendency. To evaluate NLS, we first extended ndnSim, the popular simulation tool [8] for NDN, to support label switching by adopting the MPLS extension [9]. An NLS node was created by installing NDN stack upon an MPLS node and implementing a certain forwarding policy. Then, a linear topology shown in Fig. 4 was created with channel model of *PointToPointChannel* for all links. For NLS, ILSP and DLSP were statically set up. Different Interests with prefix /cq/mpg were sent from node S, identified by automatically incremented sequence number as suffixes. The node D played the role of producer and replied with random content immediately after it received the Interests. The delivery process was simulated in both scenarios. Some simulation parameters were given below: total Interests, 100, 000, Interest sending rate, 8,000 packets per second and NDN routing protocol, *flooding*.

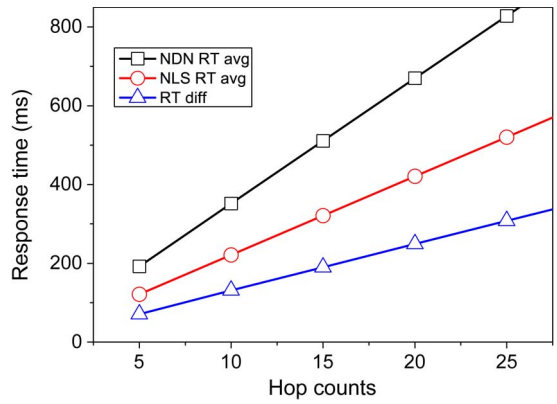

Fig. 5. Comparison of response time in native NDN and NLS by simulation.

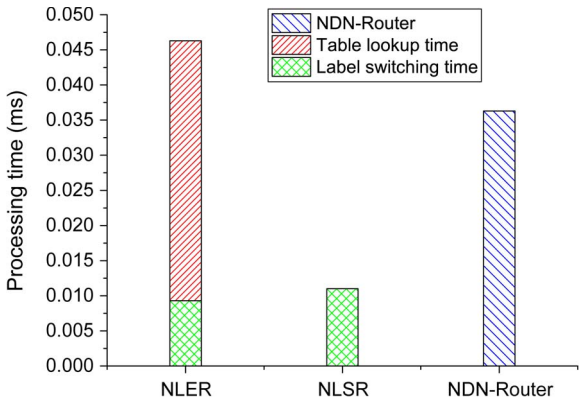

Fig. 6. Comparison of processing time in native NDN and NLS routers by simulation.

Average response time was calculated and the relationship of the *RT* dependent on the hop counts was illustrated in Fig. 5. As shown, NLS (red circle) was able to save response time by over 37% in comparison with native NDN, and as the hop counts increase, the time saving was linearly increased as expected.

To gain insight into the processing of name labels and compare its performance with native NDN name processing, we decomposed the procedure and evaluated the processing time, which is defined as the time interval from the first bit of the packet entering the node to its leaving. The simulation scenario was the same as the evaluation of response time.

The comparison results of NLS nodes with native NDN routers were demonstrated in Fig. 6. According to the figure, the average processing time of Interest packets at NLS core nodes (NLSRs, the middle diagonal square grid bar) was only 30.3% of that at NDN routers (the right bar), mainly due to the speed advantage of label swapping at NLSR over several times of name lookup at NDN routers. That meant about 70% processing time could be saved by using NLS. Besides the average processing time at the NLS edge nodes (NLERs) was about 4.2 times that at NLS core nodes, consumed mainly by name lookup in CS, PIT and LIB (red twill box). Nevertheless, the overall forwarding performance was significantly improved by using NLS core since the number of edge nodes was much less than core ones.

IV. Conclusion

To implement fast scalable forwarding for named based networking, an MPLS-like label switching paradigm, denoted by Name Label Switching (NLS), was first proposed in this letter. It features with fixed-length name label switching and edge data caching so that both single lookup latency and times of name lookup can be reduced in transit nodes, especially for a large scale network. Its principle and node architectures were presented. Its forwarding performance was analyzed and simulated in terms of data response time and Interest packets processing time. The simulation results showed that about 37% of the data response time and about 70% of the Interest packets processing time at core nodes can be saved by using NLS, compared to native NDN.

In future research, based on NLS, many features of MPLS, like traffic engineering, quality of service (QoS) and dynamic LDP, will be further investigated. And also, the effect of NLS under random requests needs to be further verified.

References

[1] T. Moors, "A critical review of 'end-to-end arguments in system design,"' in *Proc. IEEE ICC*, 2002, vol. 2, pp. 1214–1219. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=997043

[2] L. Zhang *et al.*, "Named Data Networking (NDN) Project," Univ. California, Los Angeles, CA, USA, Tech. Rep., 2010. [Online]. Available: http://named-data.net/publications/techreports/

[3] V. Jacobson *et al.*, "Networking named content," in *Proc. 5th Int. Conf. Emerging Netw. Exp. Technol.* New York, NY, USA: ACM, 2009, ser. CoNEXT, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[4] M. Varvello, D. Perino, and J. Esteban, "Caesar: A content router for high speed forwarding," in *Proc. 2nd Edition ICN Workshop ICN*, New York, NY, USA, ACM, 2012, pp. 73–78. [Online]. Available: http://doi.acm.org/10.1145/2342488.2342505

[5] W. Quan, C. Xu, J. Guan, H. Zhang, and L. A. Grieco, "Scalable name lookup with adaptive prefix bloom filter for named data networking," *IEEE Commun. Lett.*, vol. 18, no. 1, pp. 102–105, Jan. 2014.

[6] Y. Wang *et al.*, "Namefilter: Achieving fast name lookup with low memory cost via applying two-stage bloom filters," in *Proc. IEEE INFOCOM*, 2013, pp. 95–99.

[7] *Multiprotocol Label Switching Architecture*, IETF RFC 3031 Std., Jan. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3031.txt

[8] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," NDN groups, Tech. Rep., Oct. 2012.

[9] A. Bitbucket, ns-3-dev-mpls, Dec. 30, 2013. [Online]. Available: https://bitbucket.org/infuniri/ns-3-dev-mpls