2017 ACM/IEEE Symposium on Architectures for Networking and Communications

On Incremental Deployment of Named Data Networking in Local Area Networks

Hao Wu[§] Junxiao Shi[†] Yaxuan Wang[¶] Yilun Wang[§] Gong Zhang[‡]
Yi Wang[‡] Bin Liu[§*] Beichuan Zhang[†*]
[§] Department of Computer Science and Technology, Tsinghua University
[†] Computer Science Department, The University of Arizona
[¶] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications
[‡] Huawei Future Network Theory Lab, Hong Kong

ABSTRACT

A data-centric network architecture, Named Data Networking (NDN) has been developed to meet applications' growing demands of network efficiency and security. Considering that today's network is predominantly IP, any deployment of NDN requires careful system design to not only enable NDN but also support IP traffic at the same time. In this paper, we take the most popular local area network (LAN) technology, Ethernet, as an example to investigate incremental deployment of NDN. Assuming a local network with both NDN and IP traffic, we lay out three deployment scenarios: NDN-enabled hosts and all Ethernet switches, NDN-enabled hosts and all Dual-Stack switches (i.e., they can process both NDN and IP traffic), and a hybrid network with both Dual-Stack and Ethernet switches. We examine the technical issues involved in each scenario and propose solutions. In particular, in the hybrid scenario, we propose heuristics to optimize the placement of Dual-Stack switches. Compared with traditional Ethernet, introducing Dual-Stack switches can improve network efficiency and resiliency by utilizing all physical links, spreading traffic load over the links, and taking shorter paths, while still supporting IP-based applications.

Keywords

Named Data Networking; Local Area Network; Incremental deployment

1. INTRODUCTION

Named Data Networking (NDN) [23][10] is a data-centric architecture, in which the basic network service semantic has changed from "packet delivery" to "content retrieval." Unlike IP packets, which carry destination addresses for routers to forward to, NDN packets carry content names, which are used by NDN routers and hosts to match user requests (called *Interests*) with the actual content objects (called *Data*). In other words, while IP routers deliver packets to a particular destination, NDN routers retrieve named data from any place in the network. The potential benefits of NDN are well-documented, including data-centric security, scalable content distribution [11], mobility support [25], resilient networks [21], and so on.

While the basic NDN architecture applies to any network environment, local area networks (LANs) are of particular interest because of their prevalence on the Internet and the relatively low barrier to deployment. If running NDN is **easy** and **beneficial** to applications, NDN can be deployed in LANs with no external coordination and much less effort compared to wide-area Internet. In the long run, as more and more LANs are NDN-enabled, the deployment may grow from the network edges towards the core, bringing more benefits to applications.

At the same time, since today's network and applications are all built on top of IP, any deployment of NDN needs to be able to co-exist with IP as well. Existing work deploys NDN as an overlay using TCP, UDP, or IP tunnels [1, 5], which usually require manual configuration of the tunnel endpoints. Upon new deployment, such tunnel configurations need to be manually changed to include the newly deployed NDN nodes. Furthermore, these point-to-point tunnels limit NDN's capability to utilize the underlying broadcast media. In order to take full advantage of data-centric communications and ease the deployment process, we advocate to deploy **NDN directly over Ethernet**. We set forth the following design goals:

- **Co-existence with IP Traffic**: The network should be able to support IP traffic and applications. The common mechanisms, such as address-based IP and Ethernet packet forwarding, Ethernet's Spanning Tree Protocol (STP), MAC learning mechanism and the address resolution protocol (ARP), etc. should be able to run without any change or performance penalty.
- **Native NDN Support**: For NDN traffic, the network should provide native name-based forwarding instead of relying on overlays or tunnels. Mechanisms such as in-network caching, multicast, and forwarding strategies should all work natively among deployed NDN nodes.
- **Incremental Deployability**: NDN deployment should be carried out in a gradual fashion. The deployment

*Corresponding authors. E-mail: liub@tsinghua.edu.cn; bzhang@cs.arizona.edu

978-1-5090-6386-4/17 $31.00 © 2017 IEEE
DOI 10.1109/ANCS.2017.18

process may take years, during which both NDN and IP traffic should be supported, and the more NDN deployment, the more benefits to NDN applications.

- **General Applicability**: Given the prevalence of Ethernet, this paper focuses on the deployment of NDN in Ethernet LANs. However, we intend the principle and main approach to be a general solution that, after further research, can be extended to other network environments.

We propose *Dual-Stack switch* (*D-switch*)[1], which provides name-based forwarding for NDN traffic and address-based forwarding for conventional traffic such as IP. As a contrast, conventional Ethernet switches (*E-switch*) only support address-based forwarding, while the pure NDN switches (*N-switch*) only support name-based forwarding. Identifying NDN traffic by the EtherType field in Ethernet header, D-switches act as layer-3 switches and process these packets based on content names carried in NDN header, providing native NDN features such as in-network caching, loop-free forwarding, and multicast. Furthermore, D-switches do not need to be restrained by Ethernet's spanning tree protocol because NDN can detect and break forwarding loops. This allows NDN traffic to utilize all physical links in the LAN rather than a spanning tree. For non-NDN traffic, a D-switch acts as a layer-2 switch and forwards these packets based on destination MAC address.

In this paper, we focus on three identified scenarios for NDN deployment in Ethernet LANs: (i) NDN-enabled hosts, which may emit both NDN and IP traffic, and all E-switches, (ii) NDN-enabled hosts and all D-switches, and (iii) a hybrid network with both D-switches and E-switches. We present technical issues involved in each scenario and the corresponding solutions. The scenario of pure NDN networks with NDN-only hosts and all N-switches may be possible in small networks, but it does not present deployment issues and is not of interest to this paper.

In particular, we focus on the hybrid scenario, where the network contains both D-switches and E-switches, since: this is the most important stage of NDN deployment and it raises several interesting technical issues in how to orchestrate the two kinds of switches. In a hybrid LAN, D-switches and E-switches may be placed arbitrarily anywhere in the network. A D-switch may have neighbors of E-switches, D-switches, or a mix of both. The challenge is to design mechanisms to support name-based forwarding while co-existing with address-based forwarding within the same LAN[2]. For example, a D-switch may forward NDN frames to a link considered disabled by Ethernet's STP, because it makes forwarding decisions for NDN traffic only based on the content names without considering layer-2 protocols. If a neighbor E-switch gets the frame from the STP-disabled link, it may drop this frame, which results in conflicts between the two types of switching. More specifically, we investigate the following problems: (i) how can D-switches learn the forwarding table in layer-2 and build name-based FIB in layer-3; (ii) whether to forward the NDN packets that have previously traveled STP-disabled links back to STP links; (iii) the tradeoff between using STP-disabled links and the overhead; (iv) the cache of D-switches should selectively serve Interests to stabilize the content of FIB.

Furthermore, we investigate into the problem of how to place D-switches in such a hybrid network so that the benefits from deploying NDN will be maximized. We identify two heuristics, pair-based and connectivity-based, to optimize different performance goals. Simulations show that as deploying NDN grows, the network starts to spread traffic over more links, take shorter paths, and experience less traffic load.

The rest of the paper is organized as follows. Section 2 gives a brief background on NDN, Section 3 describes the NDN deployment scenarios in LANs, Section 4 presents the D-switch placement problem and heuristic solutions, Section 5 presents evaluation results, Section 6 summarizes related works, and Section 7 concludes the paper.

[1]D-switches can be implemented on general programmable platforms and software-based switches.

[2]Note that the NDN traffic frames still carry MAC address at layer-2 in order to travel among E-switches.

2. NDN BACKGROUND

Named Data Networking [24] is a new network architecture that falls in the Information-Centric Networking paradigm. Different from the IP architecture, NDN makes content ("what") a first-class citizen in all network functions, rather than "where" the content is located. This section briefly introduces NDN with a focus on its packet forwarding process.

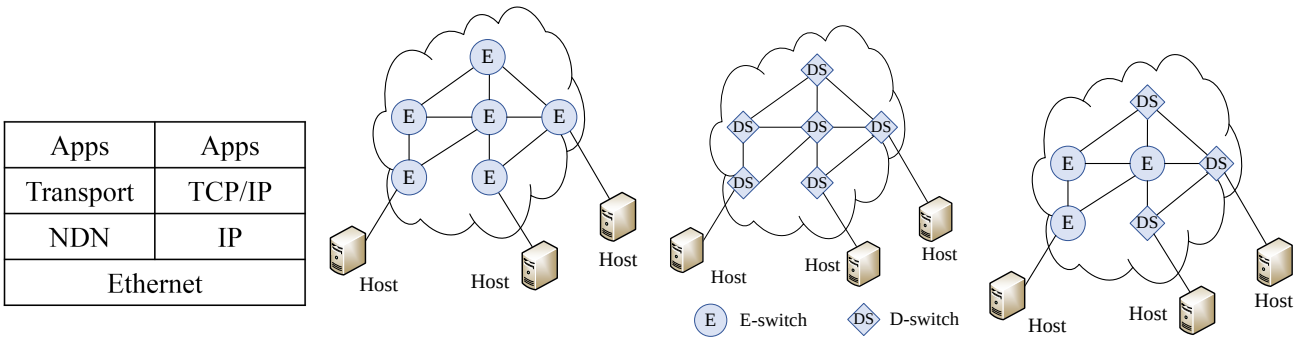
2.1 Naming

Every NDN packet carries a name that identifies a piece of content, and the packets are routed/forwarded based on their names, rather than host addresses. NDN names are assigned by applications and they are opaque to the network. A name is hierarchically structured with one or more components explicitly delimited, e.g., `/org/ancs/2017/cfp.html`. The hierarchical structure enables name aggregation and allows prefix match between names. It also provides context for consumer applications, e.g., to make trust decisions [22]. When a content changes, its name also changes, usually by including a name component for content version.

2.2 Packet Forwarding Process and Data Structures

NDN has two types of packets: *Interest* and *Data*. An NDN-enabled host that is requesting Data is called *consumer*, and the original content server is called *producer*. NDN's name-based packet processing and forwarding is very different from IP's address-based packet forwarding. Though Interests and Data are both forwarded based on their names, they are processed differently. In general, there are three data structures in NDN's forwarding pipeline: the Content Store (CS) as a data cache, the Pending Interest Table (PIT) to record interests that have been forwarded but have not received data back, and the Forwarding Information Base (FIB) to store the nexthops for each name prefix.

When a consumer wants to retrieve a piece of data, it sends to the network an Interest that carries the data name. An NDN node (e.g., a router, a switch, or a host) that receives the Interest will first look up its CS to see if it already has the requested data. If so, the data will be sent back to the Interest's incoming interface. Otherwise it will looks up its PIT to see if there is any pending Interest having the same name. If so, it means the node has requested for the data and is waiting for its return. Therefore the node will not forward the newly arrived Interest. Instead, it will



(a) The dual protocol stacks of NDN-enabled hosts (b) Scenario 1: NDN-enabled hosts and all E-switches (c) Scenario 2: NDN-enabled hosts and all D-switches (d) Scenario 3: Hybrid D-switches and E-switches

Figure 1: Deployment scenarios of NDN in Ethernet LANs

record the new interest's incoming face in the PIT entry. In this way, Interests from different consumers but asking for the same data will be aggregated in the network. If the PIT lookup is also a miss, the node will look up its name-based FIB to find out the nexthop for this Interest. It then forwards the Interest to the nexthop and record it as a new PIT entry. When the data arrives, the node will forward the data to all neighbors that have requested the data by checking incoming interface information stored in PIT. It will also cache the data in local CS. Therefore, the Data packet always takes the same path as the Interest packet, just in the reverse direction.

3. NDN DEPLOYMENT SCENARIOS IN LAN

NDN is designed as a general communication protocol that can run on any type of links. Most NDN experiments today run as an overlay over TCP or UDP tunnels for the convenience of not changing anything in existing networks. On the other hand, running NDN directly over layer-2 links, such as Ethernet, will remove the overhead of extra protocol layers and allow NDN to take full advantage of native network support.

An Ethernet LAN works on two major technical mechanisms: the Ethernet link technology implemented in each network interface card (NIC), and the self-learning and spanning tree protocol running between Ethernet switches. The NIC handles link-layer operations in receiving and transmitting frames, as well as filtering out packets whose destination address is not the host's own address. The spanning tree protocol (STP) reduces the LAN topology into a spanning tree by blocking some links, so that the self-learning process can safely flood the initial frames to the entire LAN and learn the mapping between the destination MAC address and a local switch port. To run a layer-3 protocol over Ethernet, the EtherType field in the Ethernet header needs to be set, and the upper-layer protocol needs to handle fragmentation and reassembly if a packet cannot fit into an Ethernet frame. Moreover, there must be a mechanism, such as the address resolution protocol (ARP), that maps a layer-3 address to a MAC address.

The first thing of NDN over Ethernet is to encapsulate NDN packets in Ethernet frames. This is straightforward and has been implemented in the current NDN Forwarding Daemon (NFD) [4], which sets the EtherType to NDN (`0x8624`) and handles fragmentation and reassembly when needed. The more challenging tasks are (1) which MAC addresses to use in a frame and (2) how the frames are forwarded between switches in order to retrieve the content. We will discuss these issues in three different Ethernet-LAN-based deployment scenarios. We will also touch briefly that if NDN is fully deployed and there is no legacy IP traffic, we may not even need the layer-2 MAC address, but the in-depth discussion of this topic is out of the scope of this paper.

3.1 Scenario 1: NDN-enabled hosts and All E-switches

In this scenario, the network has some NDN-enabled hosts running NDN applications (i.e., sending/receiving NDN traffic) but all the switches are conventional Ethernet switches (Fig. 1(b)). NDN-enabled hosts can send/receive either NDN or IP traffic and use EtherType to differentiate these two kinds of traffic (Fig. 1(a)), while the switches will forward all frames according to their destination MAC addresses. In this way, IP traffic is not affected at all. For NDN traffic, the question is to choose which MAC address to use.

The simplest way is to use a well-known multicast address to represent all NDN hosts. We call this multicast address the *ALL-NDN* group. All NDN traffic is sent to this address, and NDN hosts join this group to receive NDN traffic. In fact, various implementations, such as NFD [4], V-NDN [6], ndnSIM [5], have already supported this. The main benefit of this approach is that it makes the communication agnostic to endpoint addresses, consistent with the data-centric principle. When an Interest is sent, the sender does not need to pin down a particular receiver, and anyone who has the requested data can reply. The downside, obviously, is the overhead of multicasting every packet to every NDN-enabled host.

The overhead of multicasting every NDN packet can be reduced at two places. First, among multiple Ethernet segments, switches should only forward Interests to segments where Data may exist. This will require new capability in the switches, which will be developed in later scenarios. Second, once a packet arrives at an Ethernet segment, there can be multiple hosts receiving it when the underlying medium is a broadcast medium. Using the multicast address means that every NDN-enabled host in the segment needs to ac-

cept the packet and process it by the main CPU in software, e.g., matching an Interest with a local content in CS. It is much less efficient than conventional packet filtering, which filters packets based on destination addresses and is done in the network card. A recent work [17] develops NDN-NIC to enable network cards to filter out unnecessary packets based on NDN names rather than MAC addresses. We consider NDN-NIC an optional part of our deployment solution in this scenario. It improves the system performance but does not affect the correctness.

Name Prefix	MAC addr
/youtube/soccer	88:00:04:15:FE:07
/youtube/basketball	ED:05:00:06:02:11
⋮	⋮

(a) The fowarding table on consumer host

Name Prefix	Next Hop Face	
	interface	MAC addr
/youtube/soccer	3	88:00:04:15:FE:07
/youtube/basketball	1	ED:05:00:06:02:11
⋮	⋮	⋮

(b) FIB with MAC address in D-switches

Figure 2: Two types of forwarding tables with MAC address

An alternative to NDN-NIC is to use unicast MAC address in NDN packets, which will be able to take advantage of packet filtering capability in current NIC, but it will also need to dynamically learn which unicast MAC address to use for which content. This scheme is called *NDN self-learning* and the technical details are described in [18]. In short, the consumer (NDN-enabled) host maintains a forwarding table that associates a *nameprefix* with a destination MAC address, as shown in Fig. 2(a). An outgoing Interest is looked up against the forwarding table. If a match is found, the Interest packet is sent out to the destination MAC address. If not, the Interest is multicasted to ALL-NDN. When the Data packet is returned, the consumer adds a new entry recording the nameprefix and the Data packet's source MAC address and use it in forwarding subsequent packets to the same nameprefix. Essentially this scheme uses the multicast group as a discovery channel to learn the MAC addresses of content producers, but the majority of content retrieval is done via unicast.

While NDN self-learning avoids most overhead of multicasting, its drawback is that it does not take advantage of underlying broadcast medium for multicast/broadcast contents. For example, if there are multiple consumers requesting the same data, the producer will treat them as separate channels and send individual copy of the data to each consumer's MAC address. If the switches understand NDN protocol and forward packets based on content names instead of MAC addresses, they will automatically form a multicast distribution tree of the consumers. This will require new capability of the switches and is part of the next deployment scenario.

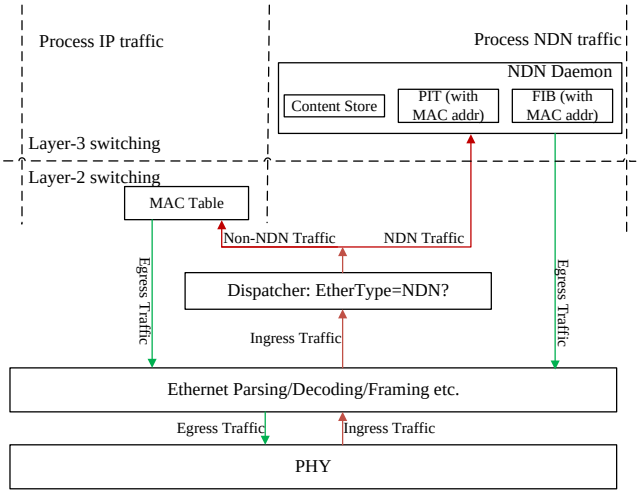


Figure 3: The architecture of a Dual-Stack switch

In summary, NDN can run directly on conventional Ethernet LANs, but to take full advantage of NDN (e.g., in-network caching and native multicast), changes like name-based forwarding and packet filtering are needed in the network, which motivates us to look into further deployment scenarios.

3.2 Scenario 2: NDN-enabled hosts and All D-switches

To provide native NDN support, we propose the concept of Dual-Stack switch (D-switch), in contrast to conventional Ethernet switch (E-switch). A D-switch is an NDN-enabled forwarding node, which is placed in the LAN and forwards NDN traffic based on content names. Fig. 3 illustrates the architecture of a D-switch. The dispatcher can identify NDN traffic by checking the EtherType field in Ethernet header. When an IP/Ethernet frame arrives, a D-switch behaves just like a regular Ethernet switch, forwarding the frame based on its MAC address. But when an NDN packet is received, the D-switch will process/forward the packet based on the content name carried in the NDN header (i.e. layer-3). For example, a D-switch maintains the CS of NDN, and if an incoming Interest finds a match in the CS, the D-switch will return the cached Data instead of forwarding the Interest. A D-switch also maintains the name-based FIB that associates nameprefixes with outgoing faces, as in Fig. 2(b). Note that the only difference between the two kinds of forwarding tables in Fig. 2 is that FIB in a D-switch has multiple interfaces. When an Interest needs to be forwarded, the D-switch will look up the content name against the name-based FIB, send the packet via the interface recorded in the matching FIB entry, and fill the destination MAC address field in Ethernet header with the recorded MAC address.

With D-switches, the LAN can support NDN features such as in-network caching and multicast. Furthermore, it actually makes the entire LAN more efficient and more resilient than conventional Ethernet LAN. We now explain these benefits assuming that **all** switches in the LAN are D-switches (Fig. 1(c)).

D-switches build their name-based FIB by NDN self-learning [18], which is the same mechanism used in Scenario 1, but runs on

the switches in addition to running on NDN-enabled hosts. When an Interest needs to be forwarded and there is no match in the FIB, the Interest is sent to all outgoing interfaces except the incoming interface where the Interest arrived. When a Data packet comes back, its nameprefix, interface and source MAC address are recorded in a new FIB entry, so that future Interests under the same nameprefix will be forwarded without flooding the neighbors. This is the efficient forwarding scheme needed in Scenario 1. A consumer does not need to know the destination MAC address of the producer; the consumer can simply send Interests to the ALL-NDN group, and D-switches will forward the Interests, based on the FIB populated by self-learning, to content producers without flooding LAN segments that do not have the content.

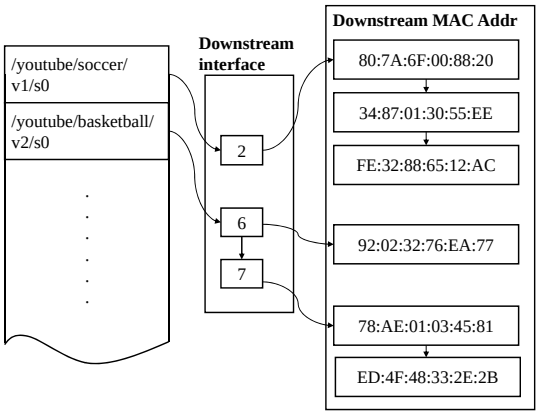


Figure 4: PIT with MAC address. A single PIT entry may associate with multiple interfaces and MAC addresses because Interests for the same Data can be aggregated.

Multicast is also supported natively. After a D-switch forwards an Interest packet, it will record the Interest and the incoming face of the Interest in PIT. Similar to FIB, PIT in D-switch needs to record the (source) MAC address as shown in Fig. 4, because NDN packets are still encapsulated in Ethernet frames in this scenario. When the corresponding Data returns, it will be forwarded to the face(s) recorded in the matching PIT entry. This assures that Data can get back to the consumer along the reverse path of the Interest. When multiple downstream consumers request the same Data, there will be only one Interest forwarded upstream but the PIT entry will record multiple downstream faces. When the Data returns, it will be sent to multiple downstream faces. This is fine-grained name-based multicast enabled by deploying D-switches.

D-switches also improve the LAN due to NDN's loop-free forwarding. Note that forwarding loops and duplicates makes flooding on an arbitrary topology prohibitive. That is why conventional Ethernet runs STP to reduce the topology into a spanning tree in order to run the flooding-based Ethernet learning protocol. In NDN, if an Interest completes a loop, it will be immediately detected because it carries the same name and *nonce* (a random number) as in an existing PIT entry, and be dropped. Therefore NDN-enabled LANs possess a significant advantage of not having to restrict NDN traffic to the spanning tree. When self-learning floods an initial Interest to the entire LAN, the learnt path will be the shortest path available in the LAN topology between the consumer and a producer; while in conventional Ethernet the learnt path is the single path allowed by the spanning tree. In this NDN LAN, no link is blocked from being used. The result is more load-balanced over links, more resiliency to failures, and shorter delays.

In summary, deploying D-switches in the network not only provides better support to NDN traffic by in-network caching and native multicast, but also makes the LAN more efficient and more resilient by utilizing all physical links available in the topology. These benefits can be the technical factors incentiving NDN deployment in Ethernet LANs.

3.3 Scenario 3: Hybrid network with both D-switches and E-switches

An interesting deployment scenario is a LAN consisting of both D-switches and E-switches, which are interconnected to forward both NDN and IP traffic (Fig. 1(d)). Our design needs to support this scenario of partial deployment since not all networks can be upgraded overnight. In fact, partial and incremental deployment is more common in real network operations than a single flag day of upgrade.

In this way, the LAN becomes a mix of traditional Ethernet switches (E-switches) and NDN-enabled Dual-Stack switches (D-switches). To support the IP traffic in such a hybrid LAN, D-switches need to run conventional Ethernet STP and Ethernet self-learning, so that all switches can build MAC-address-based switching table, and use it to forward IP traffic. However, when an NDN packet traverses the network, it will be forwarded based on its content name by D-switches (layer-3), and its MAC address by E-switches (layer-2). The challenge is to make sure that the two types of forwarding do not cause any conflict with each other. In the meantime, we also would want to provide NDN benefits even when it is only partially deployed. This is an important feature since otherwise there is no incentive for starting the deployment.

One advantage of having D-switches in the LAN is that they can use links blocked by STP since NDN forwarding is loop-free. However, in partial deployment, two additional requirements are needed to ensure the compatibility with E-switches.

- D-switch should not forward NDN packets to a neighbor E-switch across an STP-blocked link. In Fig. 5, DS1 can forward NDN packets to DS2 even the link DS1-DS2 is STP-blocked. But if DS1 forwards a packet to E3, the packet will be dropped by E3.

- D-switch must update the source MAC address of a forwarded NDN packet at each hop. In Fig. 5, for example, an NDN packet may be forwarded along the path consumer-DS1-DS2-E1-DS3-producer. Being an Ethernet switch, E1 will associate the packet's source MAC address (i.e., the consumer's MAC address if it keeps unchanged) with its incoming interface (connecting to DS2) in MAC learning, and falsely believe that sending future packets to DS2 can reach the consumer. But in reality, E1 should forward future IP packets to E2 in order to reach the consumer hosts; otherwise the packets will be dropped at STP-blocked links (because D-switches process IP packets following STP so that they will not use STP-blocked links for IP pack-

ets). This is the case that even though it works for NDN, the traffic actually misleads E-switches in their self-learning process. To remedy this problem, each D-switch should change the source MAC address of every NDN packet to its own address. In this example, E1 will only associate DS2's MAC address with interface E1-DS2, which is correct for IP traffic as well.

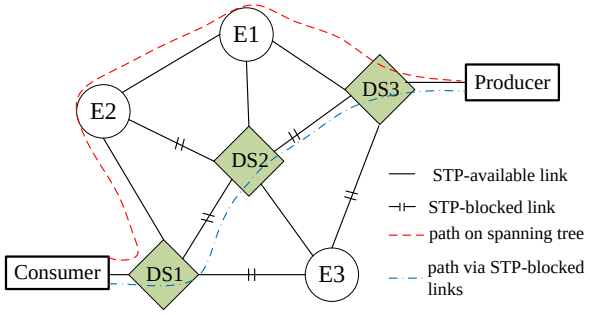


Figure 5: Packet Forwarding among D-switches and E-switches

Another important issue is whether a packet that has traversed an STP-blocked link should be allowed to be sent to an E-switch or not. Since this issue only concerns the flooded NDN packet, i.e., the discovery Interests, for ease of exposition, we name an NDN discovery Interest that has traversed an STP-blocked link as an *S-Packet*. In Fig. 5, when a discovery Interest is sent by the consumer and has been forwarded from DS1 to DS2, it becomes an S-Packet. The question is whether it should be allowed to be forwarded to E1 or not. Depending on the tradeoff, there are two options.

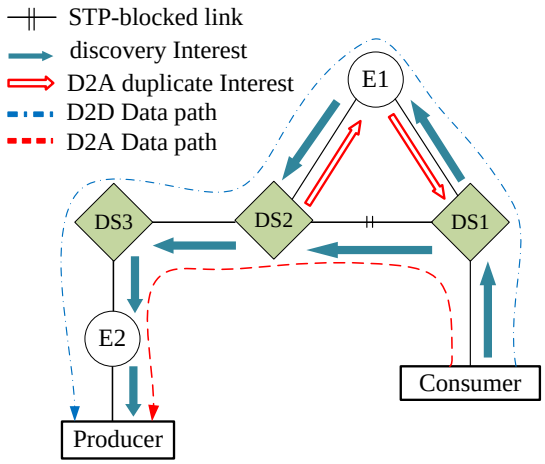


Figure 6: Flooding a discovery Interest

3.3.1 D2A: *D-switch floods S-Packets to All neighbor switches*

A D-switch floods a discovery Interest (including S-Packets) onto all links including E-switches. This allows more NDN traffic to use the blocked link at the price of some looped Interests that D-switches will eventually drop but will cause redundant traffic among E-switches.

In Fig. 6, DS1 floods consumer's discovery Interest packet to both E1 and DS2. Assuming DS2 receives the Interest first. This Interest is regarded as an S-Packet because it has traversed DS1-DS2 link which is an STP-blocked link. Under D2A scheme, DS2 sends the Interest to both DS3 and E1. As a result, two copies of the same Interest are received by E1, and the two copies will eventually be dropped by DS1 and DS2 respectively since they are detected as looped Interests. This overhead only exists during content discovery; it does not happen once on-path switches have built FIB entries for the name prefix. The benefit of this scheme is that the discovery Interest is allowed to take more links and may find better paths.

3.3.2 D2D: *D-switch sends S-Packets to D-switches only*

In this D2D scheme, a D-switch is allowed to send S-Packets only to D-switches. In Fig. 6, the reason of duplicate Interests on DS2-E1-DS1 under D2A scheme is E1 can not be able to detect duplicate Interest (i.e., same name and same Nonce). With D2D scheme, when DS2 receives an S-Packet from DS1-DS2 link, DS2 is prohibited to forward it toward E1. This conservative scheme prevents duplicate Interest copies among E-switches.

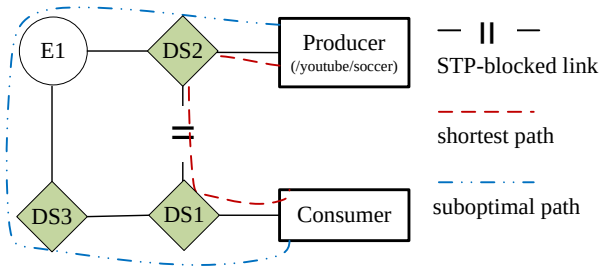
However, since the Interest on DS1-DS2 link has become an S-Packet, although DS2 can send it to DS3, DS3 cannot send it further to E2 because of the D2D constraint. Instead, DS2 must wait until receiving the Interest copy on E1-DS2 link, and send it again through DS2-DS3-E2 and finally reach the producer. Thus the price of D2D scheme is the lost opportunity of using more links.

Another rule of D2D scheme is that after a D-switch creates a PIT entry for an S-Packet, it **cannot** drop the **first normal** discovery Interest sent from a connected E-switch even if the Interest carries the duplicate Nonce. In Fig. 6, DS2 first receives S-Packet on STP-blocked link DS1-DS2 and creates a PIT entry; this Interest is forwarded to DS3 where it gets dropped. When DS2 receives the normal discovery Interest from E1-DS2 link, despite it having the duplicate Nonce and being recorded in PIT, DS2 (and DS3) must flood it again. Otherwise, none of the discovery Interests can reach E2 and the producer, which blocks the content source discovery process.

3.3.3 *D-switch does* NOT *serve discovery Interests from CS*

Each D-switch is equipped with the Content Store (CS) that can cache passing Data packets and serve the subsequent Interests requesting the same Data. However, during the content source discovery process, if a D-switch uses the cached Data in CS to serve the discovery Interest, it is possible to cause the network to learn a suboptimal path.

In Fig. 7, the consumer wants to watch a YouTube video that is partially cached at DS3; the discovery Interest is flooded by DS1 to both DS2 and DS3. Both DS3 and the producer answer this Interest with the first segment of the video, but since DS3 is closer to DS1 than the producer, DS1 would accept the Data reply from DS3 and add a FIB entry to remember that the video is available via DS3, and drop the Data reply from producer via DS2 because it arrives later. DS1 then forwards subsequent Interests for this video to DS3 according to the FIB entry. However, only a part of the video is cached at DS3; if a requested video segment is



FIB@DS3

Name Prefix	Interface	DST MAC
/youtube/soccer	E1	DS2's MAC addr

FIB@DS1 if CS@DS3 serves discovery Interest

Name Prefix	Interface	DST MAC
/youtube/soccer	DS3	DS3's MAC addr

FIB@DS1 if CS@DS3 does **NOT** serve discovery Interest

Name Prefix	Interface	DST MAC
/youtube/soccer	DS2	DS2's MAC addr

Figure 7: Network may learn suboptimal path if CS serves the discovery Interest

not in DS3's CS, DS3 would have to follow its FIB entry and forward the Interest via E1 to the producer, and the Data is returned via path producer-DS2-E1-DS3-DS1-consumer, which is longer than the shortest path through producer-DS2-DS1-consumer.

In summary, partial deployment of NDN in Ethernet LANs indeed can bring partial benefits of supporting NDN traffic natively and making use of more links. However, it needs to be carefully designed to avoid conflict between name-based forwarding and address-based forwarding at the same time.

3.4 Scenario 4: NDN-only Networks

If the NDN architecture truly takes off, we may eventually see NDN-only networks, in which all applications and traffic are NDN-based and there is no legacy IP traffic. While we cannot predict the the future of technology deployment, it is theoretically possible, especially in small, specialized networks.

In this scenario, there is no need to support IP traffic. Thus, the D-switch will be replaced by *NDN-only switch* (*N-switch*). One important implication is that now it is possible to use names as the layer-2 identifier and do away with all addresses, including MAC address. In [17], Shi et. al. explored a design for native NDN Network Interface Card (NDN-NIC), which filters incoming packets by their names rather than destination MAC address, before delivering the packets to the OS. This will make the entire network stack truly data-centric. Another point is that forwarding NDN traffic will be greatly simplified compared with Scenarios 2 and 3. N-switches are free to use any physical link since there is no STP constraint or IP compatibility to worry about. FIB and PIT in N-switches do not need to track neighbor's MAC addresses. In the producer discovery stage, the CS constraint described in Section 3.3.3 still hold in order to search the shortest path.

Table 1: Notations in Placement Algorithms

Notation	Meaning
$NodDeg\_STP(N)$	node degree in terms of STP and unlocked links: number of STP and unlocked links connected to node N
$NodDeg\_R(N)$	node degree in terms of blocked links: number of blocked links connected to node N
$LkDeg\_STP(A-B)$	link degree in terms of STP and unlocked links of link A-B: $NodDeg\_STP(A)$ +$NodDeg\_STP(B)$
$LkDeg\_R(A-B)$	link degree in terms of blocked links of link A-B: $NodDeg\_R(A)$+ $NodDeg\_R(B)$

3.5 External Traffic

Our design also supports external traffic, e.g., going to the public Internet. This type of traffic is routed by a gateway router, which is also an NDN node that can forward both NDN and IP traffic. When self-learning is used to discover the source of external contents, what will be learnt is the MAC address of this gateway router. Thus all external traffic will be sent to and received from the gateway router.

4. D-SWITCH PLACEMENT IN HYBRID NETWORK

In the hybrid deployment scenario, the network nodes consist of D-switches and E-switches. This section studies how to maximize the benefits to NDN applications in this hybrid network. In particular, given a topology and a fixed number of D-switches (the "budget"), how to place these D-switches inside the network, so as to enhance link usage, balance link load, and maximize network performance? We propose two heuristics, PDA and CDA, to solve this problem. They work for D2A and D2D forwarding schemes respectively. Notations used in these heuristics are summarized in Table 1.

The main benefit of introducing D-switches into the existing IP network is to unlock STP-blocked links, so that more links are practicable for NDN traffic. However, as described in Section 3.3, an STP-blocked link can be used only if both ends are D-switches. This means we should place pairs of D-switches on opposite ends of STP-blocked links, in order to unlock those links.

Before running either of the two heuristics, we assume all the nodes in the topology are colored as E-switches. Then we run Ethernet's STP to distinguish the STP links from the STP-blocked links. What PDA and CDA do is to color the given number (budget) of E-switch nodes as D-switches.

4.1 PDA: Pair-based D-switch Allocation

Pair-based D-switch Allocation (PDA) (Algorithm 1) selects an STP-blocked link in each iteration, and upgrades both switches connected to this link as D-switches to unlock this link. Priority is given to the STP-blocked link with largest $LkDeg\_STP$ (line 6), because unlocking this link would give the largest increment on network connectivity. After all STP-blocked links have been unlocked,

Algorithm 1 PDA: Pair-based D-switch Allocation

```
1: function PDA(topo, budget)
2:     DSwitches ← ∅
3:     blockedLinks ← findNonSTPLinks(topo)
4:     compute LkDeg_STP for each link in blockedLinks
5:     while budget > 0 AND blockedLinks ≠ ∅ do
6:         link ← link with greatest LkDeg_STP in blockedLinks
7:         remove(blockedLinks, link)
8:         for all node ∈ {link.from, link.to} do
9:             if node ∉ DSwitches AND budget > 0 then
10:                add(DSwitches, node)
11:                budget ← budget − 1
12:        compute LkDeg_STP for each remaining link in blockedLinks
13:    while budget > 0 do
14:        node ← (unselected) E-switch with largest NodDeg_STP
15:        add(DSwitches, node)
16:        budget ← budget − 1
17:    return DSwitches
```

Algorithm 2 CDA: Connectivity-based D-switch Allocation

```
1: function CDA(topo, budget)
2:     DSwitches ← ∅
3:     blockedSlices ← findBlockedSlices(topo)
4:     upgradedSlices ← ∅
5:     while budget > 0 AND blockedSlices ≠ ∅ do
6:         slice ← largest slice in blockedSlices
7:         remove(blockedSlices, slice)
8:         if budget ≥ slice.NodeAmount then
9:             DSwitches ← DSwitches ∪ slice
10:            budget ← budget − slice.NodeAmount
11:            add(upgradedSlices, slice)
12:        else if blockedSlices ≠ ∅ then
13:            nextSlice ← largest slice in blockedSlices
14:            if budget ≥ nextSlice.NodeAmount then
15:                pick budget nodes according to LkDeg_R from slice and add to DSwitches
16:                return DSwitches
17:        else
18:            pick budget nodes according to LkDeg_R from slice and add to DSwitches
19:            return DSwitches
20:    slice ← largest slice in upgradedSlices
21:    pick budget nodes adjacent to slice with greatest NodDeg_STP and add to DSwitches
22:    return DSwitches
```

if there is remaining budget of D-switches, PDA upgrades the unselected E-switch nodes with largest $NodDeg\_STP$ as D-switches (line 13-16), because they have the best connectivity and putting a Content Store at these locations is likely to bring the maximum benefits.

PDA works best with D2A forwarding scheme (Section 3.3.1), which does not restrict flooding of discovery Interests that have traversed an STP-blocked link (S-Packets). PDA may not work well with D2D forwarding scheme, which restricts S-Packets only to be flooded to D-switches. The reason is that, under limited budget, PDA prefers to place pairs of D-switches between richly connected nodes to unlock links, but the unlocked links spread all over the topology. Under D2D scheme, as soon as a discovery Interest traverses one of these unlocked links, it becomes an S-Packet and cannot be flooded further unless the neighbor happens to be a D-switch.

4.2 CDA: Connectivity-based D-switch Allocation

Connectivity-based D-switch Allocation (CDA) is designed to work with D2D forwarding scheme. The basic idea is to maximize the area reachable by S-Packets by placing D-switches close to each other.

Before we explain the algorithm, we define *Blocked Slice*, which is a subset of the topology completely connected by STP-blocked links.

Definition 1. Given the topology $L = (V, E)$ whose STP-blocked link set is $L.E_{BLK}$, a subset $BS = (V, E)$ of L is the Blocked Slice if the following conditions are met:

1. $\forall u \in BS.V, \exists v \in BS.V, makes < u, v > \in BS.E$ *and* $< u, v > \in L.E_{BLK}$.
2. $\forall u \in BS.V, \nexists v \in \complement_{L.V}^{BS.V}, makes < u, v > \in L.E_{BLK}$.

A network may have multiple Blocked Slices isolated by STP links.

Under D2D forwarding scheme, an S-Packet can only travel inside a Blocked Slice. CDA (Algorithm 2) starts with the largest Blocked Slices that can be fully upgraded within the available budget, and upgrades all nodes in those Blocked Slices to D-switches (line 8-11). If this kind of slices does not exist or there is budget left, then CDA turns to pick the smallest Blocked Slice that is just over the remaining budget, and upgrade as many nodes as budget allows (line 15 and 18). Priority is given to nodes adjacent to links of greatest $LkDeg\_R$, but CDA also requires the node to be connected to an already upgraded D-switch via a blocked link in order to avoid creating new isolated islands. Finally, in case there is still left budget after unlocking all STP-blocked links, the remaining budget is used to expand the area of the largest upgraded Blocked Slice: the unselected E-switch nodes adjacent to this Blocked Slice are upgraded as D-switches (line 20-21).

5. EVALUATION

We evaluate PDA and CDA algorithms through simulation in two real-life topologies. By comparing performance in terms of amount of practicable links, link load, Internet producer load, content retrieval latency, and packet overhead, we observe CDA is beneficial in small-scale networks with less STP-blocked links, while PDA works better in a network with abundant STP-blocked links.

5.1 Simulation Setup

We use two real network topologies including a campus network and an AS topology [2] [3], as shown in Table 2. The simulation takes two steps. First, we run PDA and CDA algorithms on a topology with a given budget (number of D-switches), or randomly draw a set of D-switches with the same budget. Second, we setup a packet-level simulator according to the topology with D-switches whose locations are determined by the first step, and evaluate network performance with synthesized traffic.

In packet-level simulations, we consider 2000 content objects whose popularity follows Zipf distribution [9]. Considering contents are more centralized in small-scale networks,

[3]We use AS701 to mimic a LAN, even though it is not a LAN topology.

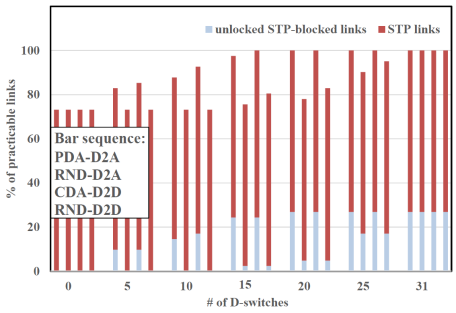


Figure 8: % of practicable links (campus)

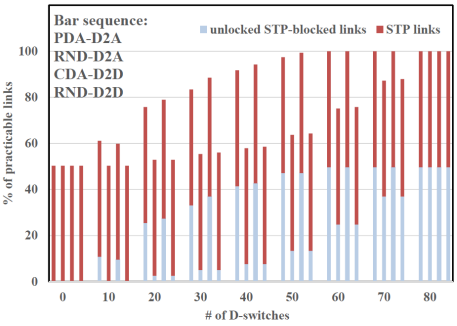


Figure 9: % of practicable links (AS701)

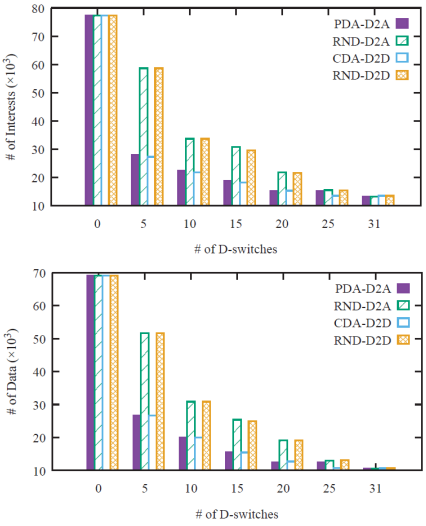


Figure 10: Average link load (campus)

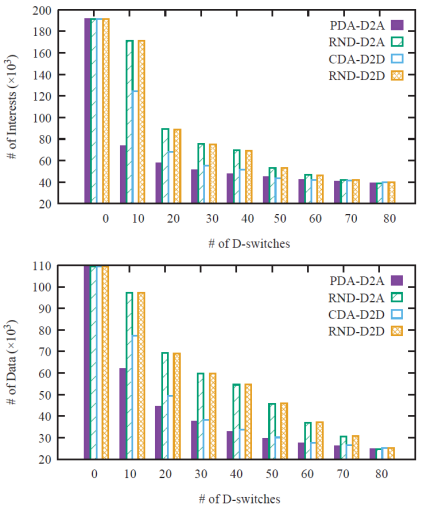
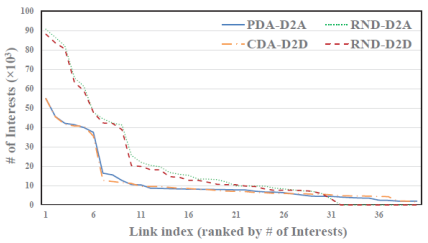


Figure 11: Average link load (AS701)



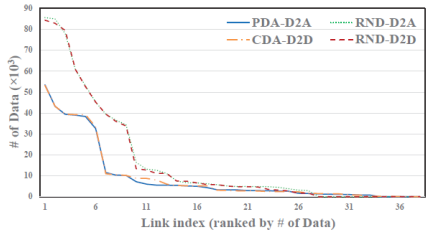
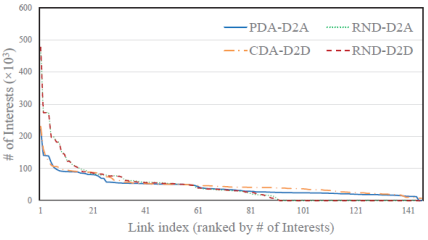


Figure 12: Link load distribution (campus, 15 D-switches)



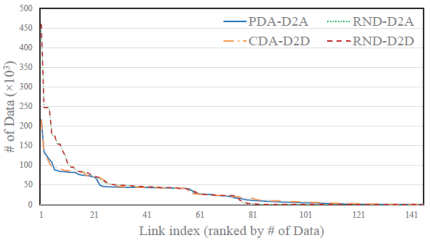


Figure 13: Link load distribution (AS701, 40 D-switches)

Table 2: Network Topologies in Simulation

Topology	campus	AS701
# of switches	31	80
# of links between switches	41	157
# of STP-blocked links	11	78
# of end hosts	14	31

the parameter of Zipf distribution is set to 1.4 for simulations on campus topology, while it is set to 0.9 for simulations on AS701 topology. Half of the content objects are uniformly located among end hosts within the topology. Each content is served by only one producer. The other half of the content objects are served from outside producers connected to one of the switches via an Internet gateway node.

Each D-switch has a Content Store (CS) that applies Least Recently Used (LRU) replacement policy; unless otherwise indicated, the CS capacity is 100 Data packets. End hosts and the Internet gateway do not have CS. Each end host in the topology has a consumer (daemon). In each simulation, the total number of Interests expressed by a single consumer is uniformly distributed in $U[30000, 50000]$. The simulator uses *slot* as the time granularity. In each slot, the switch processes all packets received in the last slot. Each packet is processed either in NDN daemon or according to Ethernet forwarding table, and it is either forwarded (moved one hop towards its destination) or dropped. A simulation terminates when all requested Data arrive at the consumer.

We compare four schemes in our simulations:

PDA-D2A PDA placement with D2A forwarding scheme.

CDA-D2D CDA placement with D2D forwarding scheme.

RND-D2A random D-switches placement with D2A.

RND-D2D random D-switches placement with D2D.

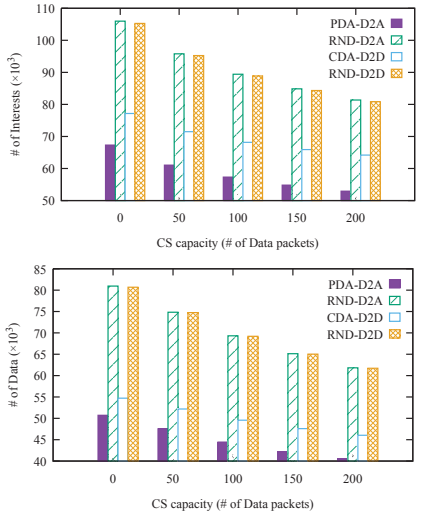


Figure 14: Average link load w.r.t. CS capacity (AS701, 20 D-switches)

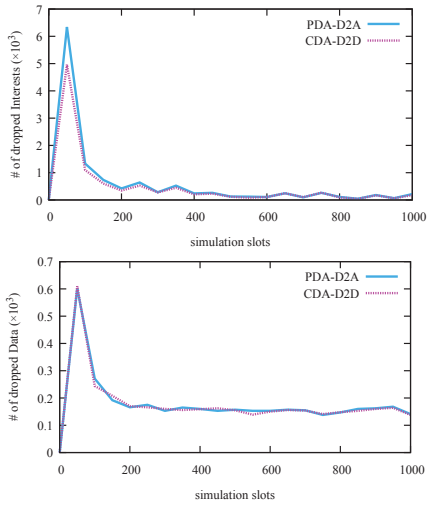


Figure 15: Packet overhead over time (campus, 10 D-switches)

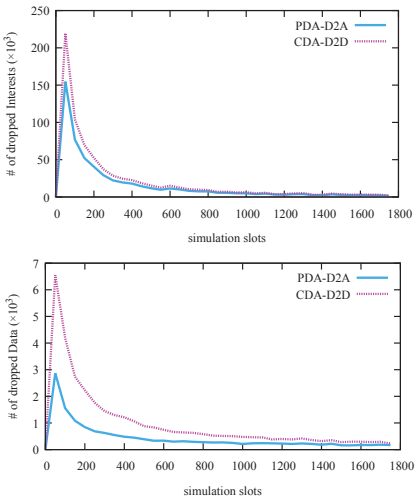


Figure 16: Packet overhead over time (AS701, 30 D-switches)

We do not evaluate PDA-D2D or CDA-D2A because these combinations are not designed to work together. For RND-D2A and RND-D2D, we evaluate five random location sets of D-switches, and report the average.

5.2 Practicable links

As mentioned in Section 3, the main benefit of introducing D-switches is to unlock STP-blocked links, making more links available to carry NDN traffic. Fig. 8 and Fig. 9 show how many links are practicable for NDN traffic, with different budget of D-switches. The practicable links come from original STP links and the unlocked STP-blocked links. If the budget decreases to 0 (zero point on X-axis in Fig. 8 and Fig. 9), the LAN degrades to the "All E-switches" scenario in Fig. 1(b), where a link is available only if it is on the spanning tree. In AS701 topology, only half of the links are on the spanning tree, resulting in a huge waste of potential link bandwidth.

The amount of practicable links increases rapidly even with a small budget of D-switches. For example, in Fig. 9, with only 25% D-switches, CDA-D2D enables near 80% links for NDN traffic. This is an obvious incentive for incrementally deploying D-switches rather than waiting for a flag-day upgrade.

In both topologies, almost all STP-blocked links are unlocked with PDA-D2A or CDA-D2D under 50% budget, but random placements have very limited benefits unless given a large budget.

5.3 Link load and traffic concentration

As we add D-switches to the network to unlock STP-blocked links, the extra links can offload NDN traffic from the spanning tree. This benefits not only NDN traffic, but also IP traffic even though they are not directly served by STP-blocked links, because a large portion of NDN traffic is moved onto STP-blocked links, so that there is less traffic competing with IP traffic on spanning tree links.

As shown in Fig. 10 and Fig. 11, average link load (including STP and unlocked links) drops significantly when we introduce D-switches. With only 15% budget, PDA-D2A is able to achieve more than 50% link load reduction on both topologies. This reduction comes from both the extra practicable links and the addition of caches that shortens data delivery paths. Random placements also achieve modest link load reductions, but as we can see in Fig. 14, PDA-D2A and CDA-D2D outperform random placements in all CS capacity settings. In some sense, the link load reduction of random placements mainly comes from the caches, because there is hardly any reduction when CS capacity is set to zero.

We take a deeper look at how traffic spread among practicable links in Fig. 12 and Fig. 13. We can see that with the same budget, PDA-D2A and CDA-D2D spread traffic onto more links than random placements. The proposed schemes' distribution curves are more smooth compared with the random ones, resulting in a lower peak level traffic among links. This reduces the risk of a single link getting congested. In the smaller campus topology, a large portion of the traffic is still concentrated in top 20% links; in the larger AS701 topology, traffic is more evenly distributed among all practicable links. There is no a major difference between PDA-D2A and CDA-D2D in this metric.

5.4 Packet overhead

Packet overhead in the hybrid network is mainly caused by the flooded discovery packets in the "cold start" period. We trace the packet overhead during a single simulation run, and plot the number of dropped packets per five slots in Fig. 15 and Fig. 16 [4]. We use 10 D-switches in campus topology and 30 D-switches in AS701 topology, considering other metrics are very close between PDA-D2A and CDA-D2D under these settings.

In campus topology, PDA-D2A and CDA-D2D are only able to unlock 6 and 7 STP-blocked links, respectively, so there is not a major difference on network-wide connectivity. PDA-D2A incurs more packet overhead because flooded Interests reach more links and are more likely to get dropped

[4]Limited by space, we cut off the curves from where they become smooth.

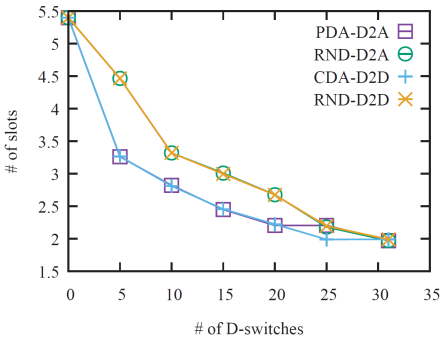


Figure 17: Content retrieval latency (campus)

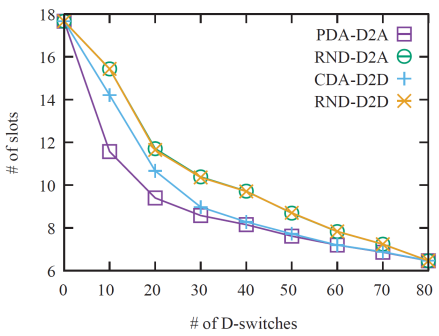


Figure 18: Content retrieval latency (AS701)

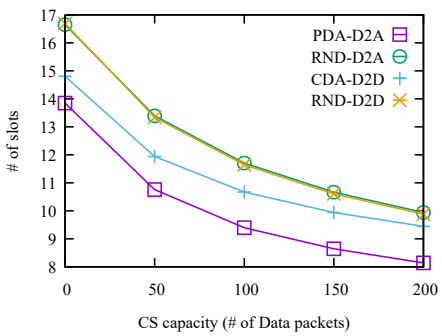


Figure 19: Content retrieval latency w.r.t. CS capacity (AS701, 20 D-switches)

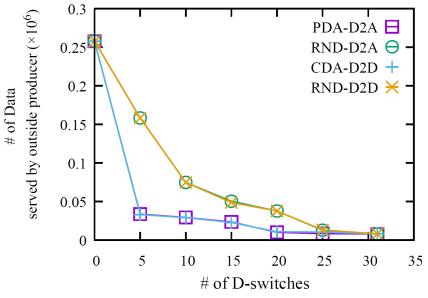


Figure 20: Internet contents served by outside producer (campus)

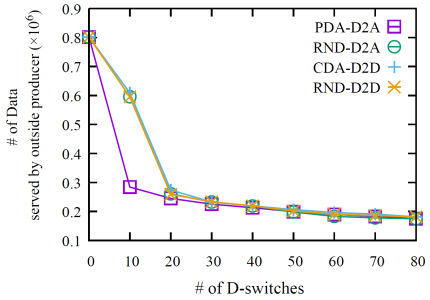


Figure 21: Internet contents served by outside producer (AS701)

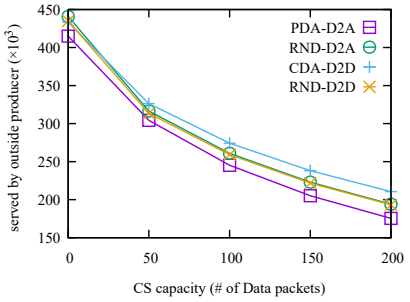


Figure 22: Internet contents served by outside producer w.r.t. CS capacity (AS701, 20 D-switches)

due to duplicate Nonce.

In AS701 topology that has abundant redundant (STP-blocked) links, PDA-D2A achieves stronger network-wide connectivity than CDA-D2D because S-Packets can be forwarded to larger extent with the help of E-switches, which allows PDA-D2A to find shorter paths. With CDA-D2D, a lot of S-Packets are dropped when they reach the boundary of the island consisting of connected D-switches and cannot be forwarded further.

Data drops are typically caused by Interest flooding: after flooding an Interest, the PIT entry is removed when the first Data packet comes back. If more than one Data come back, they will be dropped. The trends of Data drops are similar to that of Interest drops, since both occur only during content discovery.

5.5 Content retrieval latency

Content retrieval latency is an important metric that directly affects user experience in realtime applications. In Fig. 17, Fig. 18, and Fig. 19, we count the average delay for downloading a single Data packet (from either the producer or a Content Store) in terms of simulation slots.

We can see that PDA-D2A and CDA-D2D only need a small budget of D-switches to cut the latency by half. Random placements are also effective, to a lesser extent on this metric. In Fig. 19, we also see that CS capacity has significant impact on content retrieval latency.

5.6 Internet producer load

Requests for external contents can be satisfied from D-switches' Content Stores. This not only reduces traffic on the Internet access link, but also decreases content retrieval latency for external contents. In addition, this is more important for ISPs to cut down the inter-ISP charging [8, 14]. In Fig. 20, Fig. 21, and Fig. 22, we present the number of Data packets served by outside producers via the Internet access link. It is evident that, for Internet contents, the effectiveness of in-network caching is highly dependent on the amount of caches (D-switches) in the network and the CS capacity, but insensitive to where those caches are placed. Thus, the four curves in Fig. 22 are very close to each other.

5.7 Discussion

In campus topology, PDA-D2A and CDA-D2D have similar benefits in terms of practicable links, link load, and content retrieval latency, because there are few STP-blocked links in this topology, so both schemes have same link resources and CS capacity after all STP-blocked links have been unlocked. PDA-D2A has more packet overhead because Interests are flooded onto more links, so they are more likely to get dropped due to duplicate Nonce.

In the larger AS701 topology which has numerous STP-blocked links to be unlocked, PDA-D2A outperforms CDA-D2D in all evaluated metrics, especially under a small number of D-switches. CDA-D2D has more dropped packets because S-Packets are constrained by the area of connected D-switches, and they will be dropped after reaching the boundary of those connected areas.

Comparing the performance of PDA-D2A and CDA-D2D between two topologies, we get the following insight: Given a fixed topology and budget of D-switches, CDA-D2D is preferred if it is able to utilize all STP-blocked links (e.g.,

the campus network), since it has less overhead (dropped packets). Otherwise, PDA-D2A should be deployed because of the stronger connectivity (e.g., AS701).

As shown, the packet overhead incurred by flooding discovery packets sharply decreases to negligible within 5% of the simulation slots (On average, each single simulation terminates after 45000 slots). So it will not become the bottleneck in hybrid network.

Finally, the addition of Content Stores is an important factor for improving the performance of hybrid network. Even with randomly placed D-switches, cache-related metrics (overall link load, content retrieval latency, and Internet producer load) get remarkable improved. Performance gap between PDA-D2A/CDA-D2D and random placements are much smaller in cache-related metrics than in other metrics.

6. RELATED WORKS

The NDN implementation supports sending and receiving Ethernet frames, but currently there is no consideration of incremental deployment because NDN is still in its early stage.

Most existing works [5, 1] prefer to run NDN as an overlay network on top of the traditional IP network. In this way, NDN packets get encapsulated into UDP or TCP packets and spread in tunnels. However, the overlay-based implementations are still bound with UDP/TCP end-to-end sockets, which constrains NDN native features such as identifying and retrieving data from in-network caches. Although they are ideal as demos, the scale NDN network needs to deploy raw NDN stacks in the network layer.

In current NDN Forwarding Daemon (NFD) [4], NDN can run natively above Ethernet. But the only supported destination MAC address in Ethernet header is multicast address, like ALL-NDN (group) address mentioned in Section 3.1. That is, every NDN packet has to be multicast to all NDN-enabled hosts, which incurs huge overhead. On one hand, NDN-NIC [17] is an eligible patch for hosts to filter out the overhearing packets. On the other hand, to avoid multicast, FIB with unicast MAC address and NDN self-learning mechanism can be introduced in NFD design, as described in Section 3.1.

As for other NDN deployments, [20] proposes to implement pure NDN architecture within a closed local area network and use the gateway node to bridge it with outside networks. The LAN gateway is responsible to translate NDN protocol layer communication to an IP network protocol layer. Given all traffic traversing the gateway needs to be translated between the two types of protocols, the gateway is likely to become the bottleneck. Instead, we propose to use the Dual-Stack switches and NDN-enabled hosts to process NDN and IP traffic separately to avoid hot spots. [3] presents a controller-based NDN architecture to improve the delivery of Online Social Networking (OSN). An SDN-like central controller is introduced to manage switch FIBs. The challenge lies in ensuring update consistency in terms of multiple parts within a single route. V-NDN [7] is another example of running NDN directly in layer-2. It applies NDN to vehicular networks and runs over WiFi links, but it is not a hybrid network carrying on both NDN traffic and IP traffic.

The drawbacks of Ethernet's use of spanning tree have been well documented, and several alternative designs have been proposed in the IP architecture to address the problems. The most notable one is IETF standard TRILL [16], which runs a link-state routing protocol between the switches to prevent loops and allow the use of all available links. Recently, a new layer-2 design AXE [15] is proposed, in which fast failure recovery is supported. Our design takes advantage of NDN's loop-free packet forwarding, and does not require any additional control protocol among the switches.

To the best of our knowledge, this work is the first to consider incremental deployment of NDN in (Ethernet) LANs.

7. CONCLUSIONS

This paper examines the technical issues involved in deploying NDN in LANs where NDN and IP traffic are coexisting. We mainly describe three evolution scenarios, and propose the concept of Dual-Stack switch which can provide native NDN features and can be incrementally deployed with increasing benefits as the deployment grows. We further present two D-switch allocation heuristics to achieve maximum benefits under different forwarding schemes and network profiles. Through simulations, the proposed strategies have been proven to be effective and scalable. While Ethernet is the main focus of this work, the idea of incrementally deployable native NDN support and the techniques used in this work may be applied to other types of local area networks, such as WiFi networks and data center networks.

Currently, both PDA and CDA utilize LRU policy for CS replacement. As part of our future work, we plan to consider more cache management policies which are proposed in our previous works [14, 19, 12, 13]. As more impact factors, such as content popularity and download latency, are integrated into the allocation schemes, the performance is expected to be enhanced.

8. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful feedback. This material is based upon work supported by Huawei Innovation Research Program (HIRP), NSFC (61602271, 61373143, 61432009), China Postdoctoral Science Foundation (No. 2016M591182), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20130002110084), NSF Grants No. 1345142, 1513505, and 1629009. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

9. REFERENCES

[1] NDN Testbed. https://named-data.net/ndn-testbed/.
[2] UUNET ALTERNET AS701. http://bgp.he.net/AS701.
[3] ICN Research Group. Named data networking for social network content delivery, 2015. https://tools.ietf.org/html/draft-truong-icnrg-ndn-osn-00.
[4] A. Afanasyev et al. NFD Developer's Guide. Technical Report NDN-0021, Revision 6, Mar 2016.
[5] A. Afanasyev, I. Moiseenko, L. Zhang, et al. ndnsim: Ndn simulator for ns-3. *University of California, Los Angeles, Tech. Rep*, 2012.
[6] G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang. Vanet via named data

networking. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 410–415, April 2014.

[7] G. Grassi, D. Pesavento, L. Wang, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang. Vehicular inter-networking via named data. *SIGMOBILE Mob. Comput. Commun. Rev.*, 17(3), Nov. 2013.

[8] M. Hefeeda and B. Noorizadeh. On the benefits of cooperative proxy caching for peer-to-peer traffic. *IEEE Transactions on Parallel and Distributed Systems*, 21(7):998–1010, July 2010.

[9] M. Hefeeda and O. Saleh. Traffic modeling and proportional partial caching for peer-to-peer systems. *IEEE/ACM Trans. Netw.*, 16(6), Dec. 2008.

[10] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *CoNEXT*, 2009.

[11] Y. Kim and I. Yeom. Performance analysis of in-network caching for content-centric networking. *Computer Networks*, 57(13):2465 – 2482, 2013.

[12] J. Li, B. Liu, and H. Wu. Energy-efficient in-network caching for content-centric networking. *IEEE Communications Letters*, 17(4):797–800, April 2013.

[13] J. Li, H. Wu, B. Liu, Z. Fang, S. Zhang, and J. Shi. Rbc-cc: Rbc-based cascade caching scheme for content-centric networking. *Journal of Network and Systems Management*, pages 1–22, 2016.

[14] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong. Popularity-driven coordinated caching in named data networking. In *Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '12, pages 15–26, New York, NY, USA, 2012. ACM.

[15] J. McCauley, M. Zhao, E. J. Jackson, B. Raghavan, S. Ratnasamy, and S. Shenker. The deforestation of l2. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 497–510, New York, NY, USA, 2016. ACM.

[16] R. Perlman. Rbridges: transparent routing. In *INFOCOM*, 2004.

[17] J. Shi, T. Liang, H. Wu, B. Liu, and B. Zhang. NDN-NIC; Name-based Filtering on Network Interface Card. In *ACM ICN*, 2016.

[18] J. Shi, E. Newberry, and B. Zhang. On broadcast-based self-learning in named data networking. In *IFIP Networking*, 2017.

[19] H. Wu, J. Li, Y. Wang, and B. Liu. Emc: The effective multi-path caching scheme for named data networking. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, July 2013.

[20] Q. Xue, P. Tinnakornsrisuphap, and B. Mohanty. Named data networking in local area networks, Oct. 30 2014. US Patent App. 14/249,025.

[21] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A Case for Stateful Forwarding Plane. *Comput. Commun.*, 36(7), Apr. 2013.

[22] Y. Yu, A. Afanasyev, D. Clark, k. claffy, V. Jacobson, and L. Zhang. Schematizing trust in named data networking. In *Proceedings of the 2Nd ACM Conference on Information-Centric Networking*, ACM-ICN '15, pages 177–186, New York, NY, USA, 2015. ACM.

[23] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. Named Data Networking. *SIGCOMM Comput. Commun. Rev.*, 44(3), July 2014.

[24] L. Zhang, D. Estrin, V. Jacobson, and B. Zhang. Named Data Networking (NDN) Project. In *Technical Report, NDN-0001*, 2010.

[25] Z. Zhu, A. Afanasyev, and L. Zhang. A new perspective on mobility support. *Named-Data Networking Project, Tech. Rep*, 2013.