iGate: NDN Gateway for Tunneling over IP World

Ran Zhu†, Tianlong Li†, Tian Song*
†*School of Computer Science and Technology*
**School of Cyberspace Science and Technology*
Beijing Institute of Technology, Beijing, P.R.China
{zhuran_bit, tianlong.li, songtian}@bit.edu.cn

2021 International Conference on Computer Communications and Networks (ICCCN) | 978-1-6654-1278-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCN52240.2021.9522174

***Abstract*—Named Data Networking (NDN) presents strong bandwidth benefits on edge networks because its in-network caching design is probably fit for popularity and locality. However, separate NDN edge regions cannot be efficiently connected over IP backbone world, slowing down the deployment of NDN edges. Due to the difference of stateful NDN and stateless IP in forwarding principles, the coexistence of two protocols is difficult and challenging in the network layer. In this paper, we propose iGate, an NDN gateway design for tunneling over IP world, to bridge two kinds of protocols in the same layer. iGate translates protocols between stateful NDN and stateless IP by two temporary tables. It considers many factors about gateway efficiency, like table tunning, minimal NDN integration, TCP/UDP alternatives. We evaluate the performance of iGate with a real implementation on Linux and the results show that it can achieve more than 85.03% translation efficiency and less than 29% protocol conversion delay compared with IP VPN which uses the typical tunneling technology over IP network. Our work presents a practical and highly-efficient solution to assemble NDN edges into IP world, and can be applied to other similar Information-Centric Networking (ICN) designs.**

***Index Terms*—Named Data Network (NDN), Internet Protocol (IP), NDN-TCP/IP, Gateway**

I. Introduction

Named Data Network (NDN) is a promising project proposed by the US National Science Foundation under the Future Internet Architecture Program [1], and NDN is one instance of Information-Centric Networking (ICN) [2]. ICN is a more general network research direction, under which different architecture designs have emerged.

With continuous development of the network, users' demands occupy a leading position, showing a content-oriented characteristic. Being a Network architecture with a content-oriented feature, NDN is well positioned for the demand and has recently emerged with many advantages in edge network, including privacy protection, mobile content access, network traffic balance and adaptive routing [3].

Although NDN is effective in these aspects, how to evolve in the global TCP/IP dominated network is still a major challenge. There are several models for the convergence, and iGate is deployed as a bridge between NDN edges and IP backbone network. NDN is data-centric but IP network is host-centric, which complicates the coexistence.

✉ Tian Song is the corresponding author.

This work was supported by the National Natural Science Foundation of China under grant No. 92067203 and No. 61672101.

Therefore, Compatibility becomes a problem. TCP/IP backbone network plays a key role in global transmission, so it is necessary for NDN to coexist with TCP/IP backbone network for realizing global transmission. However, separate NDN edges cannot communicate with each other efficiently through TCP/IP backbone network so that the coexistence issue will be a key point in deployment. An NDN gateway is needed to forward the data, helping NDN packets to arrive at another NDN node over IP world. NDN is content-oriented which weakens the concept of host. Thus, the coexistence of two protocols presents new challenges to the network layer.

Technical challenges lie in the following aspects: First, the forwarding plane of NDN is stateful, but that of IP is stateless. The difference may increase the difficulty of protocol conversion. Especially in a more general coexistence scenario, it is a key question to preserve the states of NDN even during a long-stand transmission over TCP/IP backbone network.

Second, the unique advantages of NDN like caching may be interfered in the coexistence scenario. In NDN, duplicate requests for the same data are merged in Pending Interest Table (PIT), which means a returned Data packet can satisfy many requests according to interface ID in PIT. However, for NDN-IP gateway, how to satisfy duplicate requests from the large IP backbone network will be an issue, which may weaken the effect of in-network caching. Thus, the preservation of NDN speciality is critically important for NDN-IP gateway.

Third, realizing the structural consistency with NDN protocol is a noteworthy question to lower conversion cost for NDN-IP gateway while the preservation of NDN speciality also needs to be balanced. The internal conflict about collaborating with NDN and reducing the impact on NDN puts more challenges on the design of NDN-IP gateway. iGate is designed to improve the edge network efficiency by deploying NDN edges and IP backbone network.

For the above technical challenges and deployed NDN edges scenario, an NDN-IP gateway is designed to accomplish the protocol conversion, and our contributions are threefold:

- An NDN-IP gateway design — iGate, bridging two protocols through a tunnel over IP world, is proposed to tackle the challenges caused by semantic differences between NDN and IP packet. Moreover, iGate can cooperate with PIT and FIB in NDN to gain less conversion delay.
- A real implementation on Linux is elaborated to demonstrate the translation between stateful NDN and stateless IP. iGate works well with the kernel-level NDN and

similar ICN protocols that have been developed in our laboratory.
- TCP and UDP alternatives are designed to help iGate adapt to different network environments. Under different communication conditions and requirements, TCP/UDP performs differently and they can be freely selected.

It should be noted that this paper focuses on the design and implementation of NDN-IP gateway, and does not include the NDN protocol stack implementing details in Linux kernel, which will be discussed in our other papers. Moreover, the performance of iGate is evaluated in terms of bandwidth, conversion delay and memory usage. The evaluation parameters show that iGate achieves more than 85.03% translation efficiency, less than 30.30% conversion delay and less than 63.71% memory usage compared with IP VPN. Additionally, iGate is a Linux implementation compared with other soft routing schemes. iGate presents a practical and efficient solution to assembling NDN edges into IP world, which probably accelerates the deployment of NDN edge networks.

The rest of this paper is structured as follows: in Section II, we discuss some related work related to our work. Section III describes the detailed design of iGate, containing two key tables and the protocol conversion process on iGate. In Section IV, we evaluate the performance of iGate with a real Linux implementation and analyze the evaluation parameters, then we draw our conclusions and put forward some future work in Section V.

II. Background

NDN has many content-centric advantages innately compared with host-centric IP. IP faces the address exhaustion problem, but NDN packets identified by content do not have to worry about this. Additionally, IP is designed without privacy consideration, but NDN has a natural privacy protection characteristic. However, IP network is mature and has become the key to global interconnection. Pure NDN edges should rely on IP backbone network for global transmission. Therefore, choosing an appropriate coexistence scenario is crucial to the iGate design.

A. Deployment Scenario

Considering different deployment requirements, the deployment scenarios can be divided into 5 categories [4], which are illustrated in Fig. 1.

- IP-IP communication in NDN *ocean*.
- NDN-NDN communication in IP *ocean*.
- NDN-IP communication in NDN *ocean*.
- NDN-IP communication in IP *ocean*.
- Border Island - communication between different "islands" in separate *ocean*.

For example, the coexistence solution of IP-IP communication in NDN *ocean* pays attention to the troublesome rewrite for all IP applications in an NDN-only network. Therefore, the H2020 project POINT [5] tried to make IP applications communicate through NDN network by translating IP protocol. Similarly, Refaei et al. [6] proposed an extensible NDN-IP gateway which translates between NDN and IP packets based on pre-defined rules.

By contrast, we consider that IP backbone network under the maintenance of Internet Service Provides over past decades have become very mature; thus, changing the IP backbone is an expensive work. At the same time, NDN with in-network caching design presents strong bandwidth benefits on edge networks. Considering these unique advantages of NDN edge network, iGate realizes the edge network using pure NDN protocol. On this basis, the NDN edge networks should be interconnected without changing IP backbone network, that is, iGate enables NDN edges to communicate over IP.

B. Related work on Integration of NDN and IP

NDN-IP coexistence scheme is a hot topic. The first comprehensive survey by Conti et al. [4] classifies the existing NDN coexistence approaches. Deployment approaches introduce the NDN protocol into the TCP/IP protocol.

Overlay approach refers that NDN runs on top of the IP protocol using a tunnel over IP. On the contrary, *underlay* approach makes NDN run under the IP protocol, involving the introduction of proxies or protocol conversion gateways which are responsible for delivering and receiving requests properly. *Hybrid* approach refers to a coexistence of both IP and NDN protocols, adopting dual stack nodes to handle the semantics of IP and NDN packets simultaneously [7].

iGate belongs to *underlay* deployment due to the involved gateway. *Underlay* is also used in some previous works of NDN-IP coexistence solution:

Shannigrahi et al. proposed IPOC [8], a general purpose tunneling protocol that enables all IP applications to use NDN networks. All IP traffic generated on IP applications is forwarded via IPOC Client, which encapsulates them in NDN Interest packet and sends them into NDN network. 5G access network is implemented with NDN protocol. IPOC Gateway receives Interest packets, unpacks IP packets, and forwards them into IP backbone. As a whole, IPOC coexistence solution deploys NDN 5G mobile network on the edge of IP backbone, which is similar to the deployment scenario of iGate. However, the introduction of IPOC Client may increase the system cost and the name domain of NDN packets contains IP address, which may damage the security of NDN. In this study, iGate has not introduced new security issues, which will be discussed in Section III E in detail.

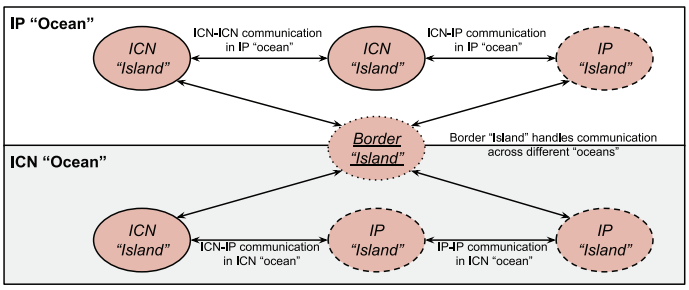

Fig. 1: Deployment scenarios for a coexistence solution [4].

In 2016, CableLabs proposed an incremental introduction of NDN in the existing CDNs to improve the content distribution [9]. The communication among them is achieved by IP tunnel. CableLabs adopts the *underlay* approach due to the gateway components, which translate IP semantics into NDN semantics. However, CableLabs relies on CCN and HTTP proxy appliances, and there are no implementations that have been conducted to clarify whether the processing requirements of these appliances will impact the cost of transition or offset the benefits of NDN. Instead, iGate is evaluated with a real implementation on Linux.

In 2017, DOCTOR [10] proposed an HTTP/NDN gateway to interconnect NDN "islands" to IP world. DOCTOR defines two types of gateways: ingress GateWay (iGW) is responsible for translating HTTP requests into NDN Interest packets and Data packet into HTTP replies, and egress GateWay (eGW) is responsible for translating NDN packets into HTTP requests. Nevertheless, DOCTOR only deals with HTTP traffic and is hard to modify to support other application protocols. By contrast, iGate can choose UDP or TCP for transmission regardless of the specific application layer protocol.

In addition to the above *underlay* coexistence solutions which are similar to the deployment scenario of iGate, Perbawa et al. proposed tap-tunnel [11], which builds an Ethernet tunnel between two nodes using NDN communication. The TAP interface is used to accept IP packets sent from IP applications and convert them into NDN packets. These NDN packets are transmitted to the other end of the tap-tunnel through NDN-testbed [12], thus, tap-tunnel realizes IP-IP communication over NDN. Additionally, The POINT project [5] in 2015 which also uses *underlay* deployment approach, enables IP-based applications to communicate in NDN. However, the two works are the opposite of our scenario.

Moreover, in the related work mentioned above, IPOC uses ndnSIM 2.0 [13] to test the IPOC protocol. DOCTOR deploys a virtual network based on OpenSwitch. Tap-tunnel uses NDN-testbed [12] to realize the NDN backbone, and the NDN-testbed enables NDN nodes to interconnect through software routing. POINT is based on an evolution of the Blackadder ICN platform [14]. In contrast, instead of simulating, iGate realizes a Linux implementation for network-layer processing in real NDN edge, which is used to evaluate the performance.

C. Challenges and Motivation

IP Internet completes packet delivery in two phases. At the *routing plane*, routers exchange routing information and choose the best routes to build the forwarding table. At the *forwarding plane*, routers forward packets strictly in accordance with the forwarding table. Therefore, IP routing is stateful while forwarding is stateless. The robust data transmission depends entirely on the routing system.

The routing in an NDN network is similar to the IP routing. The NDN routing computes routing tables which are used in forwarding the Interest packets. However, the forwarding plane of NDN includes two parts: Consumer first sends Interest packets, then Data packets are sent back along the same path in the reverse direction. Routers save the state of pending Interests to guide Data packets back to the Consumer who required the content before [15]. The *forwarding plane* of NDN is stateful, which enables each NDN router to measure packet delivery performance and make corresponding adjustments, giving NDN many special advantages including built-in network caching and multicast data delivery.

Thus, the *forwarding plane* of NDN is stateful while the *forwarding plane* of IP is stateless. Gateways are used to accomplish protocol conversion, making NDN edges connected to IP backbone. The biggest challenge is to complete the conversion between the stateful and the stateless because the forwarding state in NDN edge network will be lost in IP backbone transmission if there is no processing, which may heavily interfere the benefits of NDN. Therefore, in IP backbone transmission, NDN-IP gateway should preserve the state of NDN protocol.

Moreover, NDN-IP gateway may offset the unique advantages provided by NDN like built-in network caching. In the coexistence scenario, it is difficult for NDN-IP gateway to meet multiple Interests from the IP large network when receiving a Data packet, but it can be solved with Pending Interest Table (PIT) in pure NDN.

Additionally, NDN-IP gateway will inevitably bring extra overhead to the system, which may affect the transmission, thus, the gateway should reduce the conversion overhead and keep the structure consistent with NDN protocol stack as far as possible, that is, to design NDN-IP gateway using some important table entries in NDN protocol.

In our deployment scenario, we implement pure NDN network, and the edge networks using pure NDN protocol are connected in IP backbone. If we use the common *overlay* approach, NDN edges can not be network independent of IP, which is the opposite of our intention. Thus, we use *underlay* deployment approach to connect NDN edges into IP backbone. However, if *underlay* approach is adopted, IP backbone can not get any information about NDN edges, which motivates us to develop iGate to assemble NDN edges into IP world.

III. Gateway Design

A. Overall Architecture of iGate

Fig. 2 displays the scene for iGate. There are N NDN edges deployed at the edge of IP backbone network. Two nodes in the two edges (NDN edge 1 and NDN edge 2) communicate with each other efficiently through iGate 1 and iGate 2 across IP backbone network. Facing the challenges in coexistence scenario, two key tables are designed for iGate to process the conversion between the stateful and the stateless. One of the tables is *Gateway Translation Table* (GTT), when receiving Interest packet, GTT completes the conversion between requested data name and IP addresses. GTT actually changes content-oriented NDN packets to host-oriented IP packets. The other table is *Tunnel State Table* (TST). When iGate 2 receives the Data packet returned from NDN, TST will match the corresponding Interest request. TST actually completes the state switch between two kinds of packets in NDN. Therefore,

it can be said that GTT and TST cooperate to complete the switch between stateful and stateless.

For now, we manually configure GTT with a simple configuration file, but the prefix name and IP address matching function of GTT is very similar to the DNS system, which provides a name-to-address mapping service. This similarity gives us a new idea to improve the establishment of GTT in the future, by realizing the establishment of GTT automatically using the BIND system.

B. Protocol Conversion

As mentioned before, the biggest challenge facing NDN-IP gateway is to achieve an appropriate transition between the stateful NDN and the stateless IP. There are two kinds of protocol conversion:

One is from the stateful NDN to the stateless IP, which occurs when iGate 1 receives the Interest packet sent by Consumer or iGate 2 receives Data packet returned by Provider. This kind of conversion is actually a conversion from name-identified to IP-identified, and the IP address of both iGates needs to be determined. The other is the conversion from the stateless IP to the stateful NDN, which is much more complicated. iGate needs to cooperate with the PIT of NDN to maintain the stateful forwarding of NDN, i.e. The Data packet flows back along the same path in the reverse direction where the Interest packet is forwarded initially. We mainly use the incoming interface ID which is logically defined in PIT to maintain the state of pending Interests. Therefore, two tables are designed as follows to translate protocols between the stateful NDN and the stateless IP in iGate.

1) Gateway Translation Table (GTT): When receiving an Interest packet, GTT will complete the conversion between the requested data name and IP address. GTT actually changes content-oriented NDN packets to host-oriented IP packets. For instance, iGate 1 receives Interest packet whose name field is written in ”bit/file.txt” showing that the Consumer asks for a txt document, and iGate 1’s job is to forward the

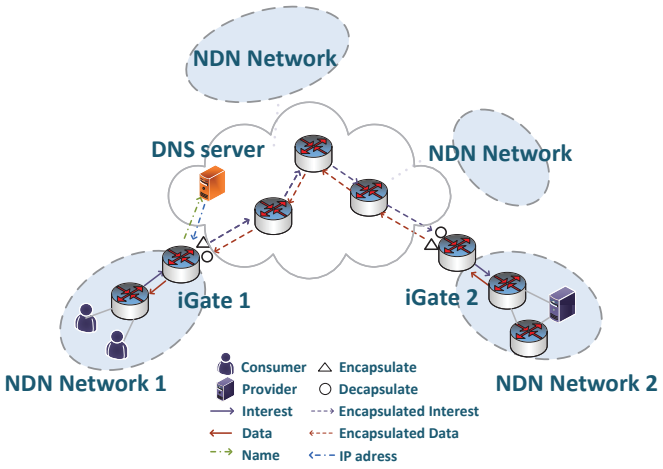

Fig. 2: NDN-IP Interconnection Scenario

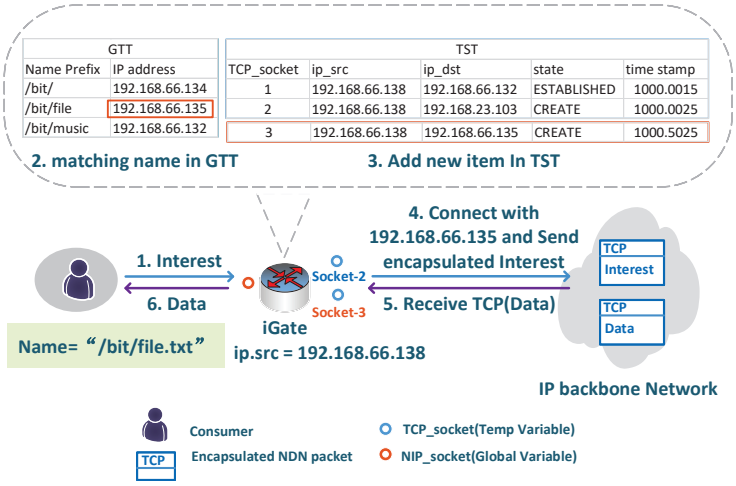

Fig. 3: Communication process on Consumer

Interest packet. Because iGate needs to send the packet to another NDN node across IP network, iGate 1 should know the corresponding IP address for encapsulating Interest packet, which can travel through IP network.

Consequently, GTT is responsible for creating a mapping between name and IP address. Owing to the fact that naming in NDN is hierarchical, an NDN edge should have the same name prefix, which offers further support for GTT. GTT uses the longest prefix matching algorithm between the name and the IP address. If iGate 1 finds the longest prefix match the IP address according to GTT, the IP address obtained will be guaranteed to be the IP address of iGate 2, which is the border gateway in NDN edge containing the Provider.

At present, we simply configure GTT with a configuration file. Fig. 3 shows an example of GTT on the Consumer side. Efforts are currently underway to automatically establish GTT, which may be realized by Domain Name System (DNS). DNS was developed in the 1980s, and the primary goal is to provide a robust and scalable name-to-address mapping service [16], [17]. As a domain name and IP address mapping, it enables more convenient access to the Internet without memorizing complex IP addresses. The process of obtaining the IP address corresponding to the host name is called domain name resolution, and the domain name is managed at three levels. DNS is named from bottom to top and inquired from top to bottom.

Compared with DNS, GTT is a mapping between data name and IP address, which has a similarity with the mapping between domain name and IP address. The name space of DNS is also hierarchical and can be thought of as a tree structure. In an analogy to DNS, the name split by ‘/’ can be resolved layer by layer in GTT, and GTT finally obtain the IP address. At the same time, we plan to use the BIND system to implement the whole establishment process of GTT. BIND is an open source software for DNS server implementation [18]. It has become the most widely used DNS server software in the world.

2) Tunnel State Table (TST): Realizing the preservation of NDN states is an issue which iGate should make a high priority to address. To achieve this goal, *Tunnel State Table* (TST) is designed, and Fig. 4 displays the detailed design of TST entries (take iGate for TCP for example). TST records quintuple information of a tunnel, two kinds of sockets, State and TimeStamp. The underlying function of GTT is to find the corresponding Interest request when iGate 2 receives a Data packet, given that iGate may receive many Interest packets which are requested from different NDN edges. TST records this correspondence using two kinds of sockets.

One socket in TST is TCP_socket, which stores the socket ID which is created by the connecting or accepting operation. Hence, TCP_socket actually keeps those tunnels beginning at iGate 1 or ending at iGate 2. The other socket is NDN_socket, which is uniquely used in NDN. An NDN_socket is created when iGate 2 receives an encapsulated Interest packet from a new tunnel, that is, NDN_socket has one-to-one relationship with the tunnel established.

TimeStamp enables iGate to know the last use time of a tunnel in real time. If iGate finds that a tunnel has not been updated within a set time, it considers that this tunnel has been lost and it will delete it. iGate will establish a new connection again and stores it in TST when receiving other requests occurring on this tunnel. TimeStamp can also demonstrate an automatic network topology recovery capability of iGate. Meanwhile, deleting TST entries according to TimeStamp can also prevent the size of TST from becoming too large.

In order to represent the tunnel state, we add "state" to TST items. Inspired by a simple coloring scheme adopted by Yi et al [15], we also use the color scheme to represent the state of a tunnel and some extra items are added. Fig. 5 displays the designed state information.

When a new tunnel is added to TST, the state of the tunnel is CLOSED, it turns CREATE when the first encapsulated Interest packet has arrived at iGate 2. A CREATE tunnel turns ESTABLISHED when the first encapsulated Data packet backs to iGate 1. By setting a reasonable time value, the tunnel state can be changed into GREEN or YELLOW. When there is no transmission through the tunnel for a long time, the tunnel state will be set as RED and be deleted in TST, showing that we have lost this tunnel.

3) Cooperate with NDN Components: In the complex transition from the stateless IP to the stateful NDN, iGate needs to cooperate with the PIT of NDN to maintain the state forwarding of NDN, helping Data packet flow back along the same path in the direction opposite to the flow direction

Tunnel State Table (TST)								
NDN_socket	TCP_socket	state	time stamp	Quintuple Information of tunnel				
				protocol	src_ip	dest_ip	src_port	dest_port
1	3	Green	100.125	1	192.168.66.135	192.168.66.138	10038	10038
2	4	Red	100.256	1	192.168.66.135	192.168.66.131	2345	2345
...	...	...	...	...	...	...	...	...

Fig. 4: Tunnel State Table (TST)

State	Description
GREEN	the tunnel can transmit data at a reasonable time.
YELLOW	the tunnel transmits data beyond a reasonable time or tunnel transmit Interest NACK.
RED	for a long time, there is no transmission through the tunnel.
CREATE	iGate 2 receives the first encapsulated Interest packet.
ESTABLISHED	iGate 1 receives the first encapsulated Data packet.
CLOSED	an initial state.

Fig. 5: The State items

of Interest. In NDN, PIT maintains the forwarding state of pending Interests, a list of incoming interfaces from which Interest packets have been received and a list of outgoing interfaces to which the Interest has been forwarded [15].

iGate uses an artful correspondence between the socket and the interface ID in NDN to solve the transition from the stateless to the stateful. When iGate 2 receives an Interest packet from a new tunnel, it will apply for an NDN_socket in the TST for this tunnel, and send the Interest packet to NDN Edge 2 via the NDN_socket. The outgoing interface in the PIT on iGate 2 is equal to the handle number of the NDN_socket. Therefore, the Data packet sent by the Provider can be forwarded back to iGate 2 in NDN Edge 2 by means of forwarding rules in NDN.

Similarly, the transition from the stateless to the stateful also happens when iGate 1 receives a Data packet returned. iGate 1 has a global variable NIP_socket, and the incoming interface ID in the PIT on iGate 1 is equal to the handle number of NIP_socket. NIP_socket is used to accept all Interest packets and the Data packets returned. Therefore, the Data packet will be forwarded along the original path and the request of the Consumer will be fulfilled.

In general, iGate realizes the structural consistency with NDN protocol, that is, use important components in NDN protocol to design NDN-IP gateway, which can lower the conversion cost. The structural consistency is reflected in two aspects: First, the information of iGate 1 will be recorded in the FIB of routers in NDN edge 1 as the default path. Thus, when the Consumer sends an Interest packet requiring for data in NDN edge 2, the data cannot be found in NDN edge 1, it will be sent to iGate 1 because iGate 1 is the default path.

Secondly, TST of iGate 2 reflects the conversion between the socket and the interface ID in NDN. Both the socket and the interface ID can uniquely identify a tunnel and are logically defined independent of the number of interfaces in the physical world. With the help of this correspondence, TST enables iGate 2 to find the corresponding Interest for the Data packet returned by the Provider, and also enables iGate 1 to find the corresponding Consumer when receiving a Data packet.

C. iGate for TCP

iGate is responsible for completing the conversion between NDN protocol and IP protocol. Under different application requirements, iGate needs to choose a suitable transport layer protocol, so iGate is designed to support the TCP protocol and the UDP protocol. iGate can flexibly select TCP or UDP protocol according to different application scenarios. TCP is a connection-oriented protocol while UDP is a connectionless

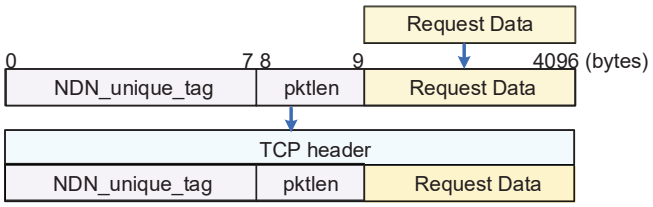

Fig. 6: Self-defined communication rule

protocol. The realization of the two kinds of systems is quite different. In this subsection, we present iGate for TCP, and iGate for UDP will be presented in the next subsection.

If NDN edges deployed on the edge of IP backbone want to communicate with each other in TCP, connecting operation will be needed first. iGate should maintain the state of each tunnel and enable the delivery and reception correctly.

1) Self-define Communication Rules: Since TCP is a streaming transmission and the streaming is border-less, the packet will be randomly divided into small packets if we send a TCP packet containing too much information. Thus, iGate should solve the issue appropriately.

To this end, we design the self-define communication rules for receiving packets correctly. An NDN packet should be encapsulated in a packet header composed of a unique tag and a length field. The unique tag "!NDNpkt!" identifies NDN packet while the length field shows the packet length in order that we can analyse the received data precisely. The data can be arranged in the structure shown in Fig. 6.

When receiving a TCP packet, iGate will firstly match the first 8 bits of the payload with the tag to figure out whether this is an NDN packet. If not, iGate will drop it because iGate only deals with NDN packets. If yes, iGate is going to locate the length field to get the size of NDN packet and then receive the packets until enough data has arrived.

2) Communication Process: The entire program includes two global sockets: ipsock and ndnsock, the life cycle of which lasts for the whole iGate running time. The whole communication process is represented in Fig. 3 and Fig. 7 from the perspective of the Consumer and the Provider respectively. The communication steps are as follows:

On NDN edge 1:

1) The Interest sent by Consumer is forwarded to iGate 1 because the content cannot be found in NDN edge 1.
2) iGate 1 matches data name in GTT to find IP address.
3) For a new tunnel, create TCP_socket and add a new item to TST. Instead, look up TCP_socket in TST.
4) Perform the connect operation with iGate 2 and send encapsulated Interest.
5) iGate 1 obtains encapsulated Data packet by monitoring TCP_sockets in TST and decapsulates it.
6) Send Data packet back to NDN via ndnsock.

On NDN edge 2:

1) iGate 2 listens at designated port and accepts via ipsock.

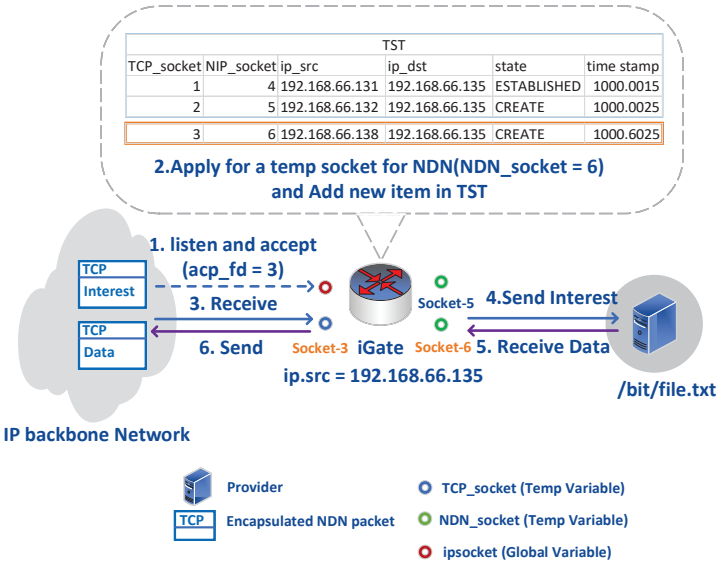

Fig. 7: Communication process on Provider

2) save TCP_socket created by accept operation and apply for a tempsocket for NDN in TST.
3) Receive and decapsulate the encapsulated Interest.
4) Send Interest packet and Receive Data packets sent by Provider via tempsock.
5) Based on the specific tempsock it comes from, iGate 2 finds TCP_socket in TST, then encapsulates and sends the Data packet.

D. iGate for UDP

Compared with iGate for TCP, iGate for UDP is much easier because UCP is a connectionless protocol. There is no connection operation between iGate 1 and iGate 2, and the packet is sent directly.

1) Communication Process: There are also two global sockets in the UDP system: ipsock and ndnsock, both lasting for the whole iGate running time. The communication steps of iGate for UDP are as follows:

On NDN edge 1:

1) Interest packet sent by Consumer is forwarded to iGate 1 because the data cannot be found in NDN edge 1.
2) iGate 1 looks up in the GTT using the requested data name and finds corresponding IP address.
3) Encapsulate the Interest packet as a UDP packet and send it via ipsock.
4) iGate 1 obtains encapsulated Data packet by monitoring ipsock and decapsulates it.
5) Send Data packet back to NDN via ndnsock.

On NDN edge 2:

1) iGate 2 receives encapsulated Interest packet via ipsock.
2) Apply for a tempsocket for this tunnel and save the number of tempsocket in TST.
3) From the tempsocket, receive the Data packet returned by Provider.

4) Based on the specific tempsock it comes from, iGate 2 finds corresponding IP address.
5) iGate 2 encapsulates the Data packet and sends it via ipsock. The Data packet flows back to the iGate who request the Data originally.

E. Discussion

1) TCP/UDP Alternatives: Network application services mainly have two choices when choosing the transport layer protocol: TCP or UDP. As a gateway bridging NDN edges and IP backbone, iGate also needs to choose the transport layer protocol. TCP is a connection-oriented and reliable transmission protocol while UDP is a connectionless transmission protocol. For different types of traffic, TCP and UDP perform differently. Therefore, iGate realizes TCP/UDP alternatives to meet different traffic requirements.

Generally speaking, UDP is suitable for these network applications running in a good network working environment or it is insensitive to the individual packet loss. These applications often have low requirements for accuracy and pursue the real-time performance, such as chat, online video and VoIP.

By contrast, for those network applications which are sensitive to the loss of packets and have a requirement for data integrity and accuracy, TCP is more suitable to be selected as the transport layer protocol, such as file transfer. However, TCP requires more system resources, and TCP needs to occupy system CPU, memory and other software and hardware resources when establishing and maintaining connections.

In the evaluation experiment presented below, it can be seen that in the laboratory environment where the network environment is good enough, the bandwidth performance of UDP is 27% higher than that of TCP on average.

TCP used to account for the vast majority of traffic on the network, but UDP traffic has increased rapidly with the development of new network services. Mena et al. [19] find that 60%∼80% of audio data streams are transmitted via UDP while TCP is used only for the control commands transmission.

As an implementation of the NDN-IP coexistence solution, iGate should pay more attention to the trend of future traffic and make a flexible choice between TCP and UDP in different traffic scenarios so as to minimize the impact of IP backbone network on NDN.

2) Basic Security Considerations: For now, our focus is not on improving the security of iGate and we just take some basic security problems into consideration. There are two basic security considerations: First, between iGate 1 and 2, the eavesdropper can intercept the IP packet and obtain the content of Interest/Data. Eavesdroppers can only know the IP addresses of two NDN edges, but the exact IP addresses of Consumer/Provider are unknown. Therefore, the eavesdropping problem of iGate is equivalent to the eavesdropping problem of NDN.

Second, fake Interest packets indeed can cause security problems. For example, Interest Flooding Attack may cause denial-of-service (DoS). A huge number of Interest packets with spoofed names sent by a malicious user can consume the bandwidth of the network and exhaust a router's memory. Focus on attacks that exploit the PIT, the huge number of Interests aim to overwhelm PIT and swamp the target content producers [20]. Once PIT is full, PIT has no memory to create entries for new incoming Interests and all subsequent incoming interests will be dropped [21]. However, in the scenario of iGate, if the tunnel uses TCP protocol, the problem is equivalent to the forged network packet injection problem of TCP protocol, and if the UDP protocol is used, it is equivalent to the injection problem of UDP protocol. Of the two situations, traditional network research has many related studies [22] [23].

From the above analysis, it is demonstrated that iGate does not bring any new security problem into network.

IV. Performance Evaluation

A. Experiments Setup

We evaluate iGate efficiency with a real implementation on Linux. With a typical Consumer–Provider model, the experimental environment of NDN edge is composed of two Industrial Personal Computers (IPCs) as iGates and two Raspberry Pi 4 Model B as the end NDN nodes, acting the Consumer and the Provider, respectively. Instead of simulating, Linux kernel is modified by us to enable the sending and receiving for Interest/Data packet, and in this way, an implementation for network-layer processing in real NDN edge is used to evaluate the performance of iGate.

Moreover, the two NDN edges are deployed on either side of the city, making the NDN edges interconnect through a large IP backbone network remotely. On the two IPCs, we also deploy PPTP VPN, which is a typical tunneling technology and also completes the transmission through IP backbone network, offering an ideal comparative object for iGate.

The experimental details are as follows: Consumer runs a burst-sending program, which sends Interest packet rapidly. Compared with ordinary sending program that the next Interest packet will be sent only when Data packet has returned, burst-sending makes full use of the network bandwidth and adopts slide window for sending. The name field in Interest packet contains the name of requested data (e.g. "bit/file.txt" to require a 10M txt document), making iGate 1 do the longest prefix match to find that this txt document is in NDN edge 2, encapsulate and send it. iGate 2 receives the request and forward, and Provider returns data to iGate 1. Then, the Interest is satisfied.

For comparison, PPTP VPN is deployed between the two IPCs and the two Pis run TCP slide window sending program asking for the same file in almost the same network environment. Moreover, MPPE encryption in PPTP is turned off to ensure the fairness in the comparison with iGate because there is no encryption on iGate.

In the rest of this section, we mainly consider three evaluation metrics on iGate: (i) *Bandwidth*: the transmission rate between the two iGates. (ii) *Conversion Delay*: additional processing time for protocol conversion on iGate. (iii) *Memory*

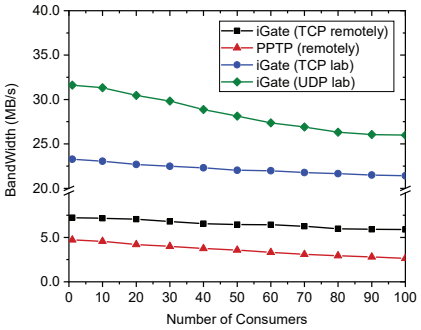

Fig. 8: Bandwidth

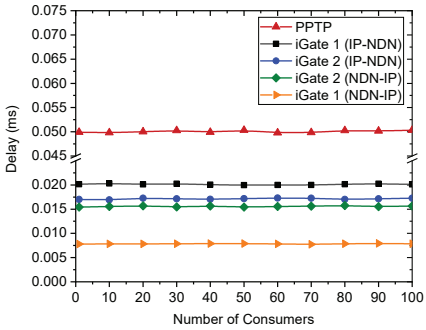

Fig. 9: Conversion Delay

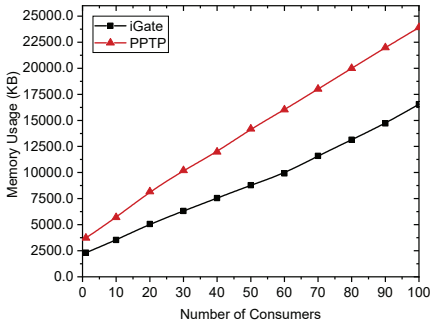

Fig. 10: Memory Usage

Usage: the memory consumed by software execution. Each of the results is the average value of 30 times.

B. Bandwidth

Fig. 8 shows the variation of bandwidth for iGate and PPTP VPN through a large TCP backbone network with the requests increase. We calculate the *bandwidth* as follows.

$$Bandwidth = \frac{10M}{t}$$

where, t is the average time for completing a 10M file transfer. In the stress test, N Consumers send the Interest packets requesting the same file concurrently. We record the transmission time for each request to be satisfied, and take the average value to get t.

For iGate using TCP tunnel, the overall communication rate is higher than 6.024 MB/s, up to 7.227 MB/s. With the increase of Consumers, bandwidth falls slightly to 84% compared to one request. In comparison, the bandwidth of PPTP VPN is from 2.645 to 4.718 MB/s and shows a marked decline to 56% compared to one request when the requests increase. With almost the same sending program requiring for the same file, iGate shows more than 85.03% translation efficiency compared with PPTP on average.

Additionally, the bandwidth condition of TCP iGate and UDP iGate are also shown to demonstrate the TCP/UDP alternatives. In a laboratory reliable network environment, UDP performs much better with 25.986-31.628 MB/s than TCP with 21.403-23.277 MB/s. Therefore, UDP tunnel will be a better choice in a reliable network.

Through such a complex and remote IP backbone network, the bandwidth of iGate for TCP is basically satisfactory, which not only shows good support for high concurrent requests, but also reflects the advantages of NDN in-network caching. The bandwidth of iGate for UDP is not measured over long-distance IP communication because under such a network environment, the high packet loss rate aggravates the bandwidth performance of iGate for UDP.

The distinct decline in bandwidth in PPTP VPN may be attributed to the fact that there is no caching in IP network, but in NDN, duplicate Interest packets will be merged in one line in PIT so that a Data packet returned can satisfy multiple requests. The slight bandwidth drop of iGate is acceptable because the transmission rate will be more easily affected by network fluctuation in a longer time caused by high traffic transmission, which is inevitable even by calculating the average values. Take 100 requests for example, the whole communication process can last for 167s, and almost every process will be affected while there are fewer requests, the performance would be much better.

C. Conversion Delay

Fig. 9 shows the delay values in a protocol conversion for iGate 1, iGate 2 and PPTP VPN. iGate works in TCP mode. *Conversion Delay* is defined as the lag time between the arrival of a packet and the completed protocol conversion. Conversion Delay is precisely calculated by clock() in the program on iGate and the timestamps of captured packets in wireshark is used to calculate PPTP VPN's delay.

iGate performs differently between NDN-IP conversion and IP-NDN conversion. IP-NDN delay on iGate 1 is around 0.02 ms and on iGate 2 is around 0.016 ms. NDN-IP delay on iGate 1 is around 0.007 ms and iGate 2 is around 0.015 ms. Compared with PPTP whose delay is around 0.05 ms, iGate performs 30.30% delay value than PPTP in a complete communication process.

From the evaluation results, the IP-NDN conversion delay of iGate 1 and iGate 2 is slightly different probably because the encapsulated Data packet is large and one IP packet may only have one piece of information of Data packet, so iGate 1 needs to receive many packets in the buffer to get a complete Data packet while the Interest packet is usually small, so it takes iGate 1 more time than iGate 2. On the other hand, IP-NDN conversion involves parsing IP packets according to the self-defined communication rules, it consumes more time than NDN-IP conversion with simple encapsulation, and thus, the delay of NDN-IP conversion is less than IP-NDN.

Moreover, in NDN-IP conversion, iGate 2 needs to match NDN_socket (storing the sockets facing to NDN intranet which is mapped with a tunnel established) in TST to obtain the corresponding tunnel information and does more computations to update the state in TST, while iGate 1 uses a global

ndnsock without matching or locating and does simple state update. Therefore, the difference between the two iGates on *Conversion Delay* occurs.

D. Memory Usage

The memory usage of iGate for TCP and PPTP VPN is shown in Fig. 10. The memory occupied by iGate increases from 2304 KB to 16555 KB while the memory occupied by PPTP VPN increases from 3721 KB to 23911 KB, both of which show an approximate linear growth. The slope of iGate curve is about 139.92, and the slope of PPTP VPN is about 202. For the size and slope of memory usage, PPTP VPN is significantly higher than iGate, and the memory usage of iGate is 63.71% less than PPTP VPN.

From the memory usage difference between iGate and PPTP VPN in the pressure test, it can be seen that iGate is lightweight and has a good support for high concurrency of iGate. Evaluation on memory usage is important for *underlay* coexistence approach because the *underlay* approach involves the introduction of proxies and protocol conversion gateways which will inevitably cause a degree of overhead.

For NDN-IP coexistence solution, a complex implementation will even damage the advantages of NDN, affecting the transfer efficiency. Therefore, an NDN-IP gateway should consider the additional overhead seriously. As a real implementation on Linux, iGate runs in the real network environment with high load, and the impact of system memory usage will be more obvious. However, the results show that iGate can efficiently run at low cost.

V. Conclusion and Future Works

NDN has many unique advantages in edge networks. However, interconnection cannot be realized directly for separate NDN edges owing to the fact that they need to communicate across IP backbone network, which may slow down the deployment of NDN edges. Considering the coexistence scenario: NDN-NDN communication in IP *ocean*, we adopt an *underlay* deployment approach to assemble NDN edges into IP world without changing the IP backbone. An *underlay* deployment approach often involves the introduction of proxies or protocol conversion gateways. Thus, iGate is proposed in this paper to solve protocol conversion between two kinds of protocols in the same layer.

An NDN-IP gateway faces many challenges and the biggest challenge is to translate protocols between stateful NDN and stateless IP, which is solved using TST and GTT in iGate. The performance of iGate is also evaluated with a real implementation on Linux, compared with PPTP VPN which also uses typical tunneling technology over IP public backbone network, and the results show that iGate can achieve more than 85.03% translation efficiency with less protocol conversion delay and less memory usage. iGate is a lightweight and practical solution to the NDN edges deployment issue. We are firmly convinced that iGate will accelerate the deployment of NDN in the IP world.

While iGate performs well in our experiment, some problems still remain to be addressed. First, we plan to realize the automatic establishment of GTT by referring to the DNS designs. Second, the security of iGate (e.g.malicious registration) will be the focus of further research.

References

[1] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. Smetters, B. Zhang, G. Tsudik, K. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, and E. Yeh, "Named data networking (ndn) project," 05 2012.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," 2009, p. 1.

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, C. Papadopoulos, K. Claffy, L. Wang, P. Crowley, and B. Zhang, "Named data networking," *Acm Sigcomm Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[4] M. Conti, A. Gangwal, M. Hassan, C. Lal, and E. Losiouk, "The road ahead for networking: A survey on icn-ip coexistence solutions," *IEEE Communications Surveys & Tutorials*, vol. PP, 04 2020.

[5] D. Trossen, M. J. Reed, J. Riihijärvi, M. Georgiades, N. Fotiou, and G. Xylomenos, "Ip over icn - the better ip?" in *2015 European Conference on Networks and Communications (EuCNC)*, 2015, pp. 413–417.

[6] T. Refaei, J. Ma, S. Ha, and S. Liu, "Integrating ip and ndn through an extensible ip-ndn gateway," in *Proceedings of the 4th ACM conference on information-centric networking*, 2017, pp. 224–225.

[7] A. Rahman, D. Trossen, D. Kutscher, and R. Ravindran, "Deployment considerations for information-centric networking (icn)," *ICNRG draft*, 2018.

[8] S. Shannigrahi, C. Fan, and G. White, "Bridging the icn deployment gap with ipoc: An ip-over-icn protocol for 5g networks," in *Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies*, 2018, pp. 1–7.

[9] G. White and G. Rutz, "Content delivery with content-centric networking," *CableLabs, Strategy & Innovation*, pp. 1–26, 2016.

[10] DeplOyment and seCurisaTion of new functiOnalities in virtualized networking enviRonnements, [Online], http://www.doctor-project.org/.

[11] M. R. Perbawa and R. F. Sari, "Performance evaluation of automatic dependant surveillance broadcast data distribution using named data networking," in *2018 2nd International Conference on Electrical Engineering and Informatics (ICon EEI)*. IEEE, 2018, pp. 1–6.

[12] ndn-testbed, [Online], https://named-data.net/ndn-testbed/.

[13] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim 2.0: A new version of the ndn simulator for ns-3," *NDN, Technical Report NDN-0028*, 2015.

[14] D. Trossen and G. Parisis, "Designing and realizing an information-centric internet," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 60–67, 2012.

[15] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, p. 779–791, 04 2013.

[16] P. Mockapetris, "Domain names-concepts and facilities," 01 1987.

[17] P. Mockapertris, "Domain names-implementation and specification," 10 1987.

[18] ISC BIND, [Online], http://www.isc.org/sw/bind/.

[19] A. Mena and J. Heidemann, "An empirical study of real audio traffic," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, vol. 1. IEEE, 2000, pp. 101–110.

[20] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "Dos and ddos in named data networking," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2013, pp. 1–7.

[21] N. Chhetry and H. K. Kalita, "Interest flooding attack in named data networking: A survey," 03 2016.

[22] A. Ramaiah, R. Stewart, P. Lei, and P. Mahan, "Preventing network data injection attacks," Sep. 26 2006, uS Patent 7,114,181.

[23] D. Boro, H. Basumatary, T. Goswami, and D. K. Bhattacharyya, "Udp flooding attack detection using information metric measure," in *Proceedings of International Conference on ICT for Sustainable Development*. Springer, 2016, pp. 143–153.