Review for iGate: NDN Gateway for Tunneling over IP World

Nitin Kumar[1], Sudipta Halder[2], and Soumodipta Bose[3]

[1] International Institute of Information Technology, Hyderabad[2021202020]
[2] International Institute of Information Technology, Hyderabad[2021202011]
[3] International Institute of Information Technology, Hyderabad[2021201086]

Abstract. NDN is data-centric, address exhaustion that happens in IPv4 and IPv6 is totally eliminated because named data is used in its place. Additionally, in-network caching significantly enhances bandwidth. When producers send data packets, the path of the interest packet is reversed and data packets are sent back to the consumers in traditional NDN [2]. When consumers send interest packets first, the routers combine multiple requests for the same packet into a single entry in the Pending Interest Table (PIT) table and forward the interest packets along the network. Separate NDN edge networks cannot be efficiently connected over IP backbone, effectively slowing down deployment of NDN edges.This research paper describes the design of a Tunnelling protocol called iGate that can be used to bridge two NDN-edge networks over IP world, along with Linux implementation. The design has been has been compared against a popular tunnelling protocol called PPTP.

1 Summary

NDN having several advantages like increased bandwidth,a protocol was proposed in this paper called iGate, which is responsible for tunneling of NDN nodes over an IP ocean and improves the edge network efficiency.

1.1 Architecture

Two NDN edge nodes can communicate over IP-backbone using iGate 1 and iGate 2. gateways [1]. The challenges faced while designing such a protocol were converting the stateful forwarding plane of NDN to the stateless forwarding plane of IP. To solve this problem two temporary tables were introduced:

- **Gateway Translation Table (GTT):** Converts the requested named data into the IP address of the iGate gateway. Since names in NDN are hierarchical like DNS, it uses the longest prefix match to find the appropriate IP of IGate2 and uses the BIND [9] system for implementation.
- **Tunnel State Table(TST):** Completes the state switch between two kinds of packets in NDN and maps the data packet to the interest packet at iGate 2. The entries in the TST consist of:

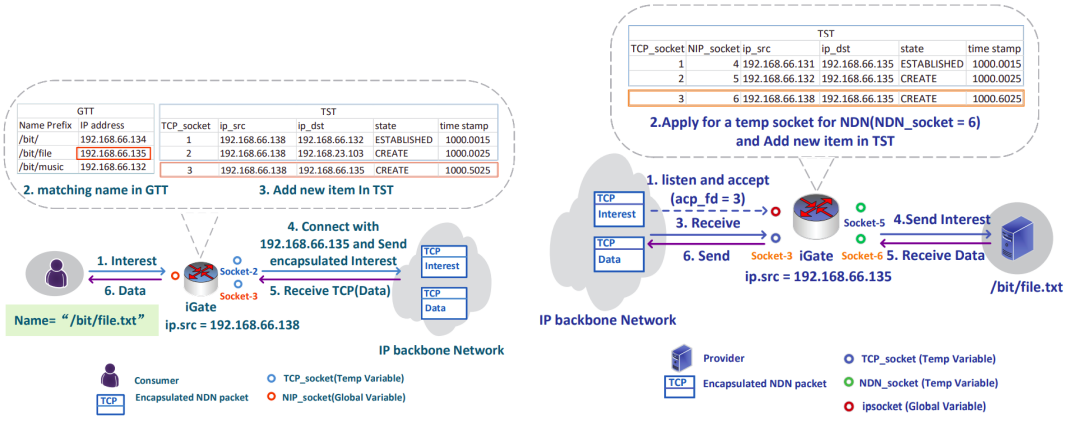
- **NDN_socket:** Socket ID that is created when iGate 2 receives an encapsulated Interest packet from a new tunnel. Each socket has a one-to-one relationship in establishing the tunnel.
- **TCP_socket:** Socket ID that is created during the connecting or accepting operation. It creates the tunnel starting at iGate1 and ending at iGate 2.
- **State:** It is used to describe the state of the tunnel to resolve various operations. These are the states:
  - **CLOSED:** A new tunnel is added to the TST table [1].
  - **CREATED:** 1st encapsulated Interest packet arrive at iGate2 [1].
  - **ESTABLISHED:** 1st Encapsulated data packet return to iGate1 [1].
  - **GREEN/YELLOW:** Based on the time set the tunnel changes between green and yellow [1].
  - **RED:** If there are no transmissions for a long time changes to RED and the entry is deleted [1].
- **Quintuple tunnel information:** Consists of protocol, source IP, destination IP, and source port and destination port which is required for establishing a connection between the two tunnel gateways iGate1 and iGate2 [1]. In the following figures, we can see the states of the NDN edge 1 (Fig. 1a) and NDN edge 2 (Fig. 1b)

1.2 Communication Process

Since iGate is only responsible for protocol conversion at the network layer depending on requirements both TCP,UDP and others can be used at network layer. TCP may encounter issues because it is a streaming protocol and because of the random partition of data into tiny packets. The solution provided by iGate is using a special tag called "ndnpkt", which uniquely identifies a packet by matching the first 8 bits of its payload with the tag. As soon as enough data has arrived, iGate will start receiving packets after locating the length field to determine the size of the NDN packet.

On NDN edge 1:

1. If content requested by consumer is not found in NDN edge 1, the Interest packet is forwarded to iGate 1 [1].
2. iGate 1 matches the data name in GTT to find the IP address of iGate 2 using the longest prefix match [1].
3. Create a TCP socket and add it as a new entry in TST table, if we are creating a new tunnel,or else look it up in TST table [1].
4. Send the encapsulated Interest packet through the IP network after completing the connect process with iGate 2 [1].
5. By keeping an eye on TCP sockets in TST, iGate 1 collects and decapsulates encapsulated Data packets send by iGate 2 [1].
6. By finding the corresponding NDN socket to the TCP socket,and send the data packet back to the consumer in edge network 1 [1].



(a) Communication process on NDN edge 1 [1]

(b) Communication process on NDN edge 2 [1]

Fig. 1: Communication process on NDN edge 1 and NDN edge 2 [1]

On NDN edge 2:

1. At the designated port, iGate 2 allows connections via the IP socket which was known by iGate 1 [1].
2. Save the TCP socket produced by the accept procedure, then use it to apply for a temporary NDN socket in TST table [1].
3. Obtain the interest packet from IP network and decapsulate it [1].
4. Forward the interest packet to NDN edge network 2 using the temporarily created NDN socket [1].
5. Receive Data packets from the Producer from edge network 2 [1].
6. iGate 2 locates the corresponding TCP socket in TST table by the already mapped temporary NDN socket. The Data packet is encapsulated and send over the IP network [1].

In contrast to TCP, UDP for iGate is much easier as it is connectionless. The bandwidth performance of UDP is 27% higher than TCP on average.

1.3 Study and Results

- iGate had 85.03% more bandwidth efficiency than PPTP VPN [1].
- The conversion delay of IP-NDN at iGate 1 is around 0.02ms and iGate 2 is around 0.016ms [1].
- The conversion delay of NDN-IP at iGate 1 is around 0.007ms and iGate 2 is around 0.015ms [1].
- iGate performs 30.30% better than PPTP in a complete communication process [1].
- Memory Usage increment: iGate(2304KB to 16555KB), PPTP(3271KB to 23911KB), showing 63.71% less memory usage [1].

2 Critiques

2.1 Assumptions

1. **Assumption:** They have initialized the entry in the GTT table Manually with the help of a configuration file,and assumed that automating it will be not much difficult and will be done using DNS Server, but things are way more complicated than they seem to be.
   Problem: Using DNS Server works in case of Domain Name and IP mapping in IP network because the number of IP address are finite (exhaustive) and also the number of corresponding mapping of Domain names is also finite, so having a finite memory at the DNS server for this mapping will do the job due to one to one mapping of domain name and IP addresses but the number of Names of Data objects possible can be very very large and thus having a finite memory will not work for this translation (also multiple names could have been mapped to same iGate Node address i.e. Many to one mapping), even if we consider that memory is not a limitation, computational complexity will still be a limitation.
   Solution: What we can do is keep this Name of Data object and the corresponding IP (of iGate[extended]) mapping cached in Content Store of iGate[Extended] in the same way we are storing frequently requested Data Objects in Content Store (we have proposed that iGate[Extended] has Content Store as well as other features of NDN Node as well as all features of IP Router), we can dedicate some part of Content Store to be used for this purpose. It will reduce the requests that the DNS Server will get and hence it will reduce burden on its DNS Server. Once this dedicated part of Content Store gets filled we will need to replace entries of the cache, so for the Caching strategy we can use algorithm like MRPGA:Multiple-Round Parallel Genetic Algorithm [6] to determine which entry to keep and which to remove in O(1) time.
   We will refer to iGate[Extended] and it's functionalities later.

2. **Problem:** iGate fails when the device (iGate Node) is moving (For example large submarines operating on NDN network) and hence they will be getting an updated IP address(since IP allocation is not static) and hence making the GTT Table entry inconsistent with current data as it has been assumed GTT table entries will be static.
   Solution: iGate should validate (Using DNS like alternative for NDN Network) the current IP address of Destination even if a match is found in the GTT Table.

2.2 Technical Approach

1. **iGate only solves NDN-NDN communication in IP ocean:** iGate mainly focuses on integrating NDN edges into the IP world [1]. But, in the

paper, there are other scenarios also which need to be taken care of [1]. They are:
(a) IP-IP communication in the NDN ocean [1].
(b) NDN-IP communication in the NDN ocean [1].
(c) NDN-IP communication in the IP ocean [1].
iGate doesn't take care of the above mentioned scenarios. If we want hassle free communication between nodes irrespective of type(NDN/IP), we need to tackle these situations also.
Solution: We can tackle the IP-NDN and NDN-IP tanslation in the following way:

We can tackle the translation challenge by addressing how to bring the NDN behaviour into the IP entity in order to semantically emulate the NDN activity. As a result, we purposefully divide the channel into two sorts of packets: interest and data channel [3].

By implementing these channels, the gateway can recognise the IP packet marked as an interest packet or a data packet in the network layer. The asymmetric name resolution table is introduced to provide a naming service for binding an IP address with a variable length prefix name that is established statically in advance. The name resolution table is made up of two separate tables: the producer table and the consumer table [3].

Fig. 2 illustrates the components of a dual-channel translation gateway. The IP interest channel is made up of a static IP address that is devoted to sending or receiving IP interest-like packets between the IP node and the gateway. The IP data channel also includes a static IP address that is dedicated to delivering or receiving IP data-like packets between the IP node and gateway. The name resolution database contains information on the prefix name associated with the IP address and port number combination [3].

Two different possibilities are examined. One scenario (Scenario 1) depicts an IP node acting as a producer and an NDN node acting as a consumer. The second shows an IP node acting as a consumer and an NDN node acting as a producer (scenario 2) [3].

Fig. 3 illustrates how the two scenarios were translated. We first assume that the gateway has a cached copy of the mapping from prefix names to a set of IP addresses and port numbers. In case 1, the gateway receives an interest packet from the NDN node with the prefix name, /abc. By finding the corresponding prefix name in the producer table, the gateway then converts the NDN interest packet to an IP interest packet. The source address header of the IP packet created by the gateway is 10.10.10.1:7777, while the destination address header is 10.10.10.17:1001. Last but not least, the gateway delivers it over the interest channel. With the destination address header set to 10.10.10.2:9000, the IP node directly responds by transmitting the content of an IP packet across the data channel [3].

For scenario 2, the IP consumer transmits a desired IP packet. The IP node creates an IP header packet with an empty data payload, 10.10.10.1:8000 as the destination address, and 10.10.10.17:1002 as the source address. The fact that the packet uses IP 10.10.10.1 as its destination address informs the gateway that it is an interesting packet when it receives it. The gateway transforms this packet into an NDN packet with the source IP address and port number mapped to a specific prefix name. The interest packet with the prefix name /eee is received by the NDN producer, who immediately responds by delivering the data packet to the gateway. By using the IP packet's 10.10.10.17:1002 and IEEE construction, the gateway transforms the packet [3].

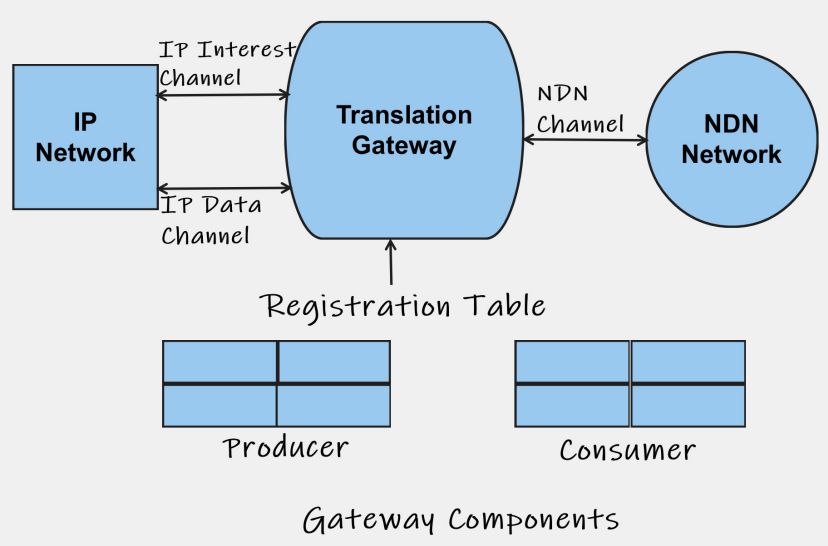


Fig. 2: Gateway components [3]

2. **Packet injection and other attacks in tunnel:** Since the entries of Tunnel State Table (TST) are not encrypted, it is very easy for the attacker to find out the src_ip and dest_ip of each entry and hence uniquely identify each tunnel. Once he uniquely identifies a tunnel, he can modify the things passing through the tunnel easily. The possible attacks are illustrated below:
   (a) Firstly, There is no system that can quickly and effectively filter every packet that has been tunneled, excluding the obviously ineffective technique of filtering all packets. Adversary can inject false packets in tunnels which will lead to false information both interest and data packet. Data can get corrupted due to wrong packet transmission.
   Solution:

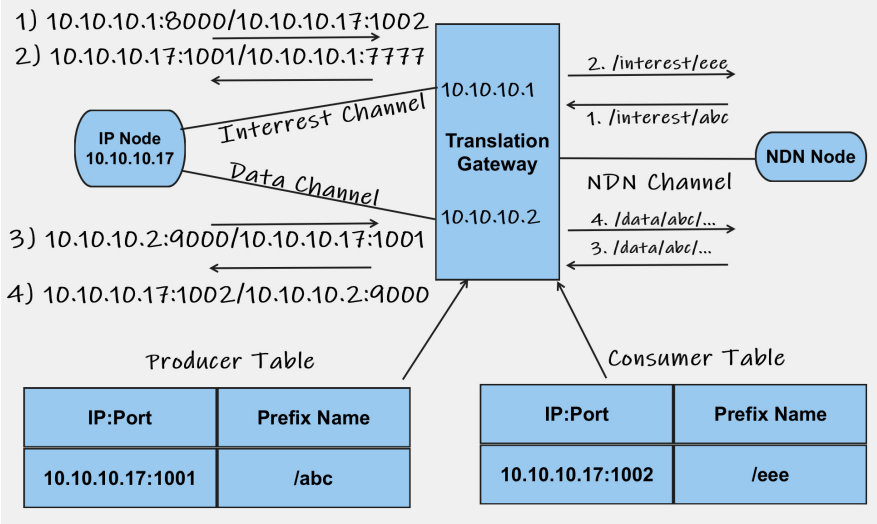


Fig. 3: Gateway components [3]

i. TST table entries should be encrypted so that attackers can't figure out tunnel IP.
ii. Inbound and outbound rules for packets should be incorporated to prevent this.

(b) Secondly, Adversary can modify packets passing through the tunnel and that would lead to wrong information at both consumer and provider end.
Solution:
i. TST table entries should be encrypted so that attackers can't figure out tunnel IP.
ii. Inbound and outbound rules for packets should be incorporated to prevent this.

(c) Third, adversary can capture packets passing through the tunnel and can extract information from them at ease. So, adversary would be able to know about the data which is being passed through the tunnel.
Solution:
i. TST table entries should be encrypted so that attackers can't figure out tunnel IP.
ii. The packets passing through the tunnel should be encrypted.

3. **Problem & Solution:** Suppose an Interest packet or Data Packet is generated inside NDN Network and it does some hops and it reaches iGate and gets encapsulated in IP packet, and for each hop in IP Network the hop count of IP Packet is updated but hop count of encapsulated NDN packet is not updated, which can cause problem such as Interest Packet or Data Packet staying active in the network for more time than it is intended to. So when an NDN packet is encapsulated in an IP packet then the Hop count of

NDN should be used as HOP count of IP packet, and during decapsulation the HOP count of IP packet should be converted back to that of NDN.

4. **Problem:** iGate has not proposed a scheme of load balancing, as it will be required in real world, there is mapping of one name to one IP address only in the GTT Table but in real world a request could be served from multiple NDN Network so that any NDN Network and corresponding iGate network is not overwhelmed by too many requests.
   Solution:
   (a) GTT Table should store mapping of name to Doubly Linked List of IP addresses of other iGate and this list will contain several information about iGate which will help decide which iGate should be contacted to get the data for the request.
   (b) Otherwise we can solve the problem in a similar manner as suggested in the paper [4] but for iGate.

5. **Problem & Solution:** Multiple NDN (say NDN2,NDN3,NDN4,NDN5,...) asking for same data object from a given NDN (say NDN1), consequence of this will be that NDN1 has to serve request for same data again and again for each request from NDN2,NDN3,NDN4,NDN5,... and the corresponding iGate of NDN1 will have to do the encapsulation of requested data again and again, instead we propose iGate[Extended] to have Content Store and other feature of NDN such as PIT, so that any duplicate request coming for same data can be directly sent by iGate[Extended] by just changing certain parameters for each reply and avoiding the overhead of requesting data from NDN1 again and again and doing encapsulation again and again. So in simple words iGate[Extended] is just an NDN Node with GTT Table and TST Table, and at another interface it has an IP Router to enable sending Encapsulated data to the IP Network. The phenomenon is described with diagrams((Fig. 4), (Fig. 5)) and solution below.

6. **Problem:** Just like NDN iGate[Extended] also suffers from DoS attack via IFA (Interest flooding attack) coming from other NDN Network.
   Solution: In order to solve this problem for iGate[Extended] we can try to reduce the memory consumption by PIT and as a result of it we end up decreasing the malicious traffic, the same way ChoKIFA: A New Detection and Mitigation Approach Against Interest Flooding Attacks in NDN has done it for NDN [5].

7. **Problem:** iGate Also brings a security vulnerability that it maintains the TST and GTT table, the content of GTT table and TST table could reveal too much information about both the NDN network and the requested data, DoS attack can be designed along with the help of cache poisoning, also intruder can join the NDN network can generate dummy data with the

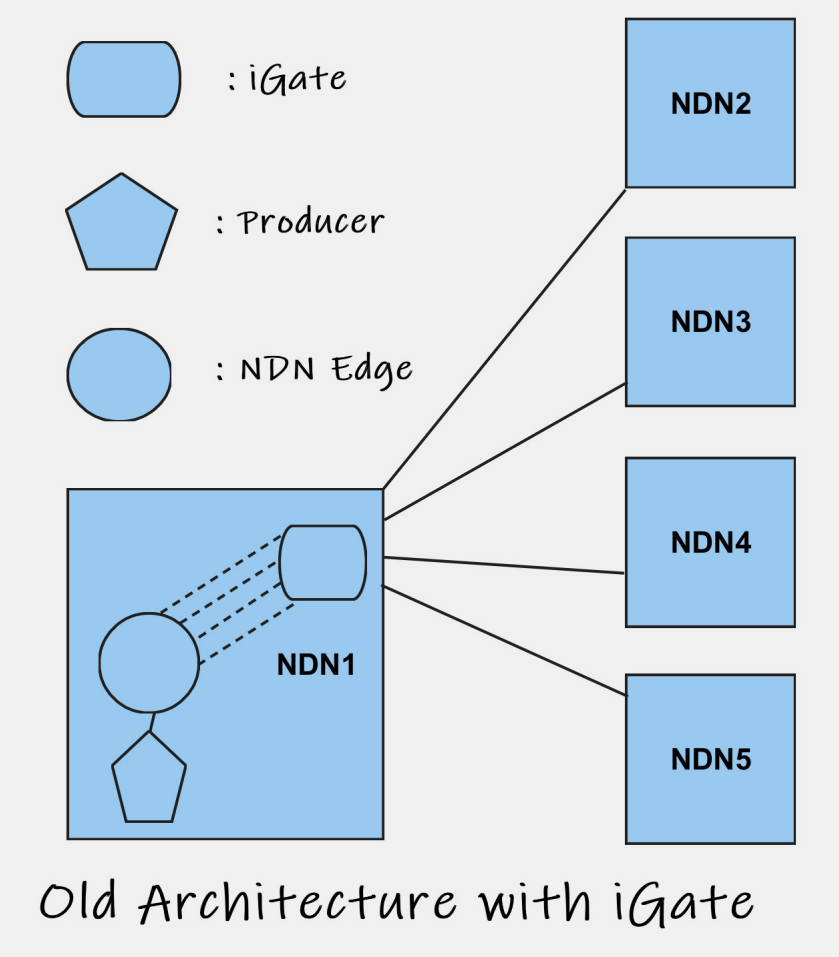


Fig. 4: Old Architechture with iGate

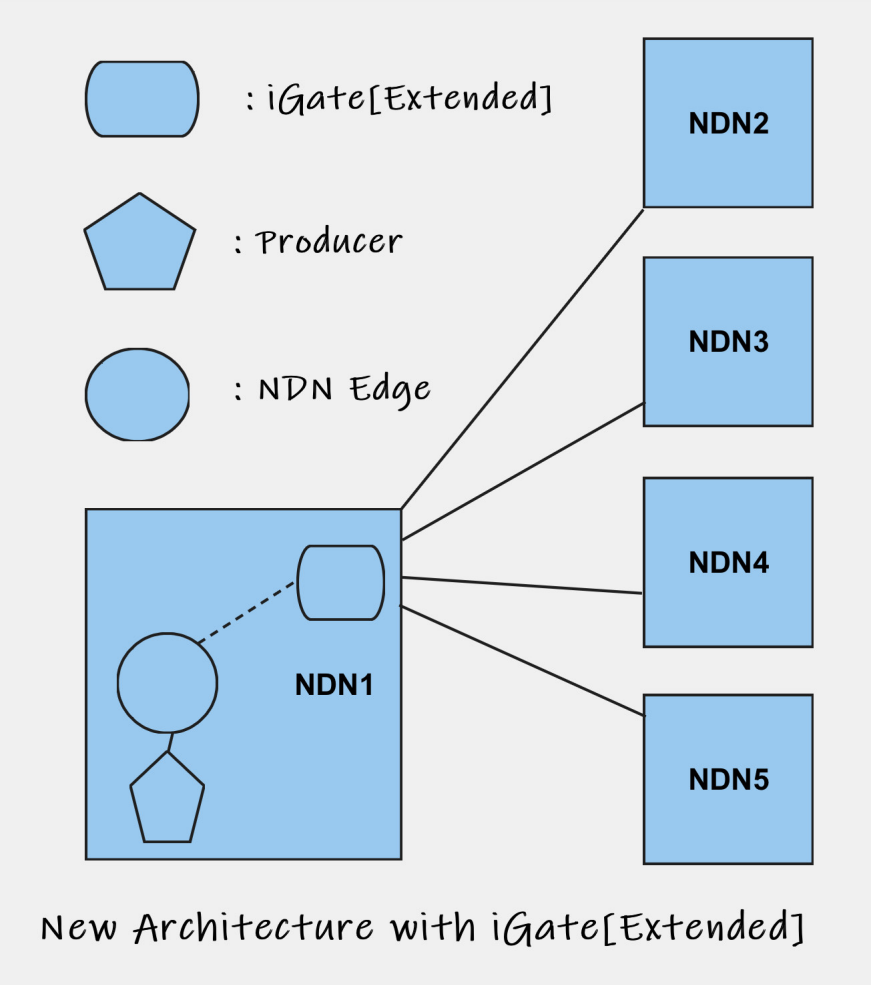


Fig. 5: New Architechture with iGate[Extended]

requested data name, although there will signature validation but this validation will be performed at the consumer end [11] and it will be too late for data validation, and the consumer will ask for the data again and again, this way the network performance could be brought down. Also sometimes the interest packet contains incomplete names [11], for those cases it is really necessary that iGate has some security measures to prevent revealing too much information.
Solution: This problem can be easily solved if we keep the information in GTT table and TST table in encrypted format or we can use the method proposed similar to the one proposed in the following paper. It focuses to device a method so that Content name itself does not reveals much information [7].

8. **Problem:** Whenever there is communication required from one NDN to other NDN and the medium of transmission between both the NDN is through IP, i.e. IP Network connects the two distantly separated NDN, then iGate Encapsulates the NDN Interest request or data reply inside and IP Packet, the problem with this approach is that the data generated at NDN has to be signed, and once this encapsulated packet enters the IP Network there it can suffer from fragmentation depending on the MTU that the channel can support or max permissible payload size and once these fragmented packet reach the destination NDN (Note: the path taken by these individual fragments could be different also, or some may not even reach the destination) it will make no sense to receiver as it will be treated as invalid data due to signature validation of received data packet as iGate is just doing encapsulation and decapsulation of Interest and Data when required and it is not doing the reassembly of received packets/fragments.
In communication when reply comes through IP Network from other NDN edge, the packet can come in many fragments so the iGate has to have the ability to assemble them and convert them from network layer fields to NDN related field of this level, this paper has not considered this in experiment setup.
That's the reason they were not able to detect the inconsistency that has arrived due to fragmentation of encapsulated packet in IP world, as they didn't had any application running on NDN edge to actually verify that the packet which has been received is valid or not as iGate is just bothered about the encapsulation and decapsulation in the two network, making iGate almost useless except for the case of igate for TCP. The phenomenon is described with diagrams((Fig. 6), (Fig. 7), (Fig. 8), (Fig. 9)) and solution below.

   Solution: iGate should have the ability of both IP router as well as an NDN router, so that iGate can reassemble the packet received from IP world and then decapsulate it and send it inside NDN edge. We call this variation of iGate as iGate[Extended], it should have the ability of header checksum, TTL validation along with the features that is expected from an NDN router

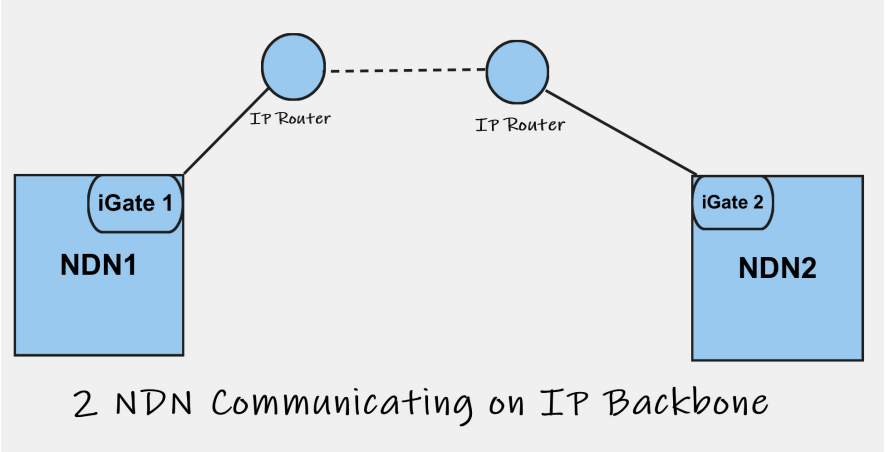


Fig. 6: NDN communicating on IP backbone

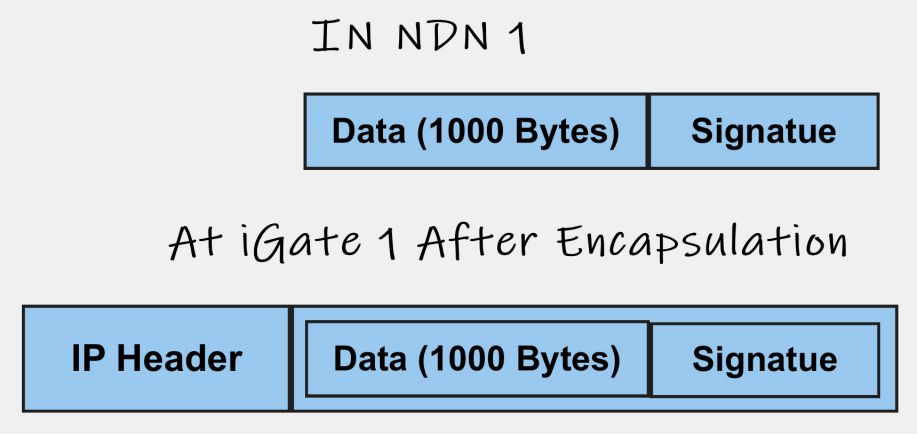


Fig. 7: At iGate1 After Encapsulation

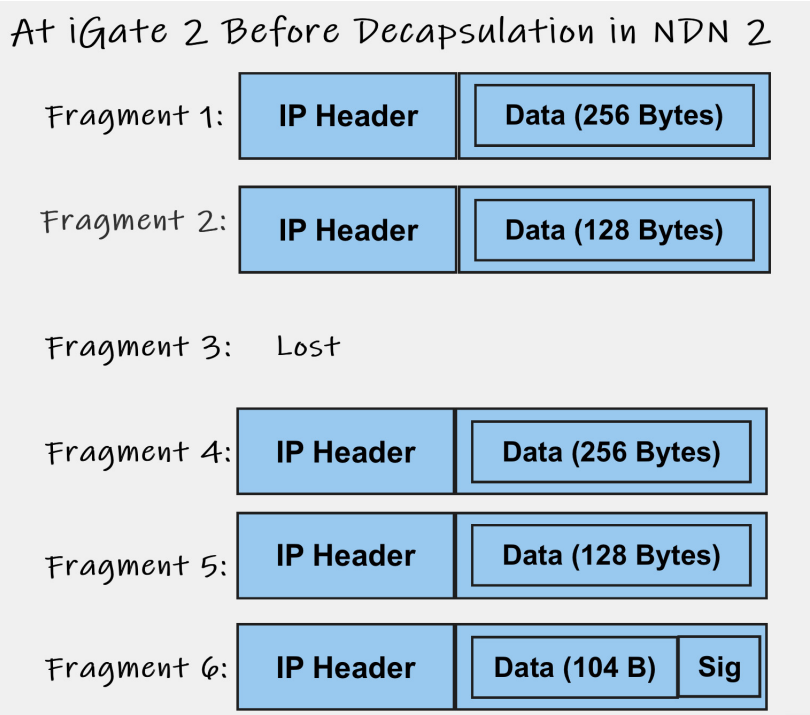


Fig. 8: At iGate2 before Decapsulation in NDN 2

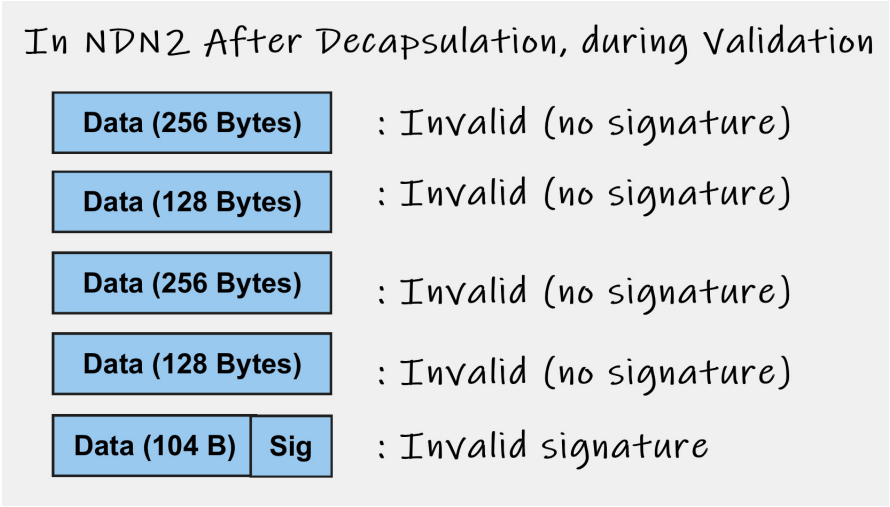


Fig. 9: In NDN2 after decapsulation, during validation

such as cache storage.

9. **Problem:** Another attack possible on the tunnels is wormhole attack. Wormhole attacks are serious and common in VANETs and other ad-hoc networks. In this attack, there are two or more malicious nodes, and the data packets are tunneled from one end to the second malicious node at the opposite point before being broadcast [8].

   Solution: TST table entries should be encrypted such that attackers can't find tunnel IP.
   We can detect and prevent wormhole attack using AOMDV protocol.

2.3 Analysis/Results

1. **Comparison of performance of iGate with backdated VPN technologies:** Here the performance of the iGate has been evaluated against PPTP VPN which uses standard tunneling technology over IP network [1]. But this comparison is not fair since PPTP is very old and vulnerable VPN technology. Since Windows 95, it has been available in various forms [10]. The performance of PPTP VPN is not as per the latest VPN technologies like OpenVPN, L2TP, SSTP. Hence, the bandwidth and conversion delay in PPTP VPN would not be as good as the latest VPN technologies. From experiment, we came to know the following two points:
   (a) On average, iGate has a translation efficiency of more than 85.03% when compared to PPTP [1].
   (b) In a complete communication process, iGate outperforms PPTP by 30.30% [1].
   (c) So, these statistics might change if we compare iGate with the latest VPN technologies like OpenVPN, L2TP, SSTP. Only then we would be able to measure the actual performance of iGate.

   Solution: We have latest VPN technologies like OpenVPN, L2TP, SSTP. We have to set up the iGATE experiment accordingly so that it aligns with the chosen VPN technology. Then, we would reevaluate the results and come to the conclusion how better or worse is the performance of the iGATE in comparison with the chosen VPN.

2. **Probem & Solution:** For comparison the tunneling technique they have used is PPTP VPN although they have disabled encryption in PPTP VPN which is fine because iGate is also not encrypting anything, but a lot of latency could have come from authentication protocol in PPTP VPN which can explain poor performance of PPTP VPN as compared to iGate in terms of conversion delay, as authentication is performed before conversion in PPTP VPN.

3 Improvements and Extensions

1. We have proposed iGate[extended] to solve most of the problems that iGate faces. iGate[extended] is an NDN node with the abilities of iGate and to be able to communicate to IP network, it has IP router at its other interface .
2. We have also proposed that even after finding a matching entry at GTT table, validation should be performed because the IP address given to iGates are not static.
3. We have also proposed a mechanism to reduce the expected load the DNS server using the dedicated part of content store[iGate extended] for storing the mapping of name and IP address of other iGate.
4. Since the paper has not solved the connectivity issue between NDN and IP network, we have proposed a conversion protocol which can convert NDN packet to IP packet and IP packet to NDN packet using dual channel translation gateway [3].
5. To prevent packet injection, packet sniffing, packet capture attack in the tunnel between two iGates, TST table entries should be encrypted, inbound and outbound rules for packts should be incorporated and the packets passing through the tunnel should be encrypted.
6. To prevent an interest packet or data packet to stay active in the network more than its lifetime, we have suggested to synchronize the hop count of NDN packet to the hop count of IP packet.
7. We have also proposed a scheme to use the GTT table in such a way that it also accounts the load on other iGate nodes and request service from nodes which have less load.
8. Since we have proposed the iGate[extended] to have content store it can avoid some redundant task such as encapsulating same data again and again(if multiple NDN network are requesting same data) by using its content store to cache the outgoing IP packet
9. We proposed to encrypt the data inside GTT table so that the name and the IP mapping does not reveal too much information about the data and location.
10. We have tried to solve the inconsistency that can come due to fragmentation of encapsulated data in IP network by giving the ability of reassembly and validation to the iGate[extended].
11. Another attack possible on the tunnel between two iGATE is wormhole attack which can be detected and prevented using AOMDV protocol.
12. Performance of the iGate has been evaluated against PPTP VPN. The performance of PPTP VPN is not as per the latest VPN technologies like OpenVPN, L2TP, SSTP. Hence, the bandwidth and conversion delay in PPTP VPN would not be as good as the latest VPN technologies.
    So, need to use the latest VPN technologies(OpenVPN, L2TP, SSTP) while measuring performance of iGate.

4 Conclusion

iGate: NDN Gateway for Tunneling over IP World has proposed a very good method to solve the connectivity issue between two NDN Network connected through IP Network and it solves some part of problem of co-existence of IP and NDN, but iGate does not solves connectivity issue between IP and NDN, also it is not scalable yet and has its limitation. We have tried to find a lot of problems that iGate will face and proposed solutions for them. Also we have proposed a variation of iGate named as iGate[Extended] and its functionalities. With our proposed solution iGate[Extended] will not only solve connectivity issues between two NDN but will also solve the connectivity issue between IP and NDN Network. And we have also proposed some mechanism to make iGate more efficient.

References

1. Zhu, R., Li, T., Song, T. (2021, July). iGate: NDN Gateway for Tunneling over IP World. In 2021 International Conference on Computer Communications and Networks (ICCCN) (pp. 1-9). IEEE.
2. Zhang, L., Estrin, D., Burke, J., Jacobson, V., Thornton, J. D., Smetters, D. K., ... Papadopoulos, C. (2010). Named data networking (ndn) project. Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC, 157, 158.
3. Fahrianto, F., Kamiyama, N. (2021, July). The Dual-Channel IP-to-NDN Translation Gateway. In 2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) (pp. 1-2). IEEE.
4. Mansour, D., Osman, H., Tschudin, C. (2020). Load balancing in the presence of services in named-data networking. Journal of Network and Systems Management, 28(2), 298-339.
5. Benarfa, A., Hassan, M., Compagno, A., Losiouk, E., Yagoubi, M. B., Conti, M. (2019, June). Chokifa: A new detection and mitigation approach against interest flooding attacks in ndn. In International Conference on Wired/Wireless Internet Communication (pp. 53-65). Springer, Cham.
6. Yang, F., Tian, Z. (2021, November). MRPGA: A Genetic-Algorithm-based In-network Caching for Information-Centric Networking. In 2021 IEEE 29th International Conference on Network Protocols (ICNP) (pp. 1-6). IEEE.
7. Leshov, N., Yaqub, M. A., Khan, M. T. R., Lee, S., Kim, D. (2019, July). Content name privacy in tactical named data networking. In 2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN) (pp. 570-572). IEEE.
8. Sharma, S., Kaul, A. (2018). A survey on Intrusion Detection Systems and Honeypot based proactive security mechanisms in VANETs and VANET Cloud. Vehicular communications, 12, 138-164.
9. ISC BIND Homepage, `http://www.isc.org/sw/bind/`.
10. `https://www.howtogeek.com/211329/which-is-the-best-vpn-protocol-pptp-vs.-openvpn-vs.-l2tpipsec-vs.-sstp/`.
11. `https://named-data.net/project/ndn-design-principles/`.