Automated Tunneling Over IP Land: Run NDN Anywhere

Arthi Padmanabhan
UCLA
artpad@cs.ucla.edu

Lan Wang
University of Memphis
lanwang@memphis.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

ABSTRACT

Named Data Networking (NDN) proposes a fundamental architectural change to the Internet, moving from point-to-point communication to a data-centric model. NDN-enabled nodes can communicate over any substrate that can deliver datagrams, such as layer-2 links (WiFi, BLE, Ethernet, etc.) and IP/UDP/TCP tunnels over IP connectivity. However in the latter case, NDN-enabled nodes must be able to discover the presence of each other and the data each serves in an automated way. This poster describes the design of an NDN Neighbor Discovery service (NDND), which enables isolated NDN nodes to discover each other and interconnect through tunneling over IP connectivity.

CCS CONCEPTS

• **Networks** → **Network protocol design**;

KEYWORDS

Named Data Networking, deployment, rendezvous

ACM Reference format:
Arthi Padmanabhan, Lan Wang, and Lixia Zhang. 2018. Automated Tunneling Over IP Land: Run NDN Anywhere. In *Proceedings of 5th ACM Conference on Information-Centric Networking, Boston, MA, USA, September 21–23, 2018 (ICN '18),* 2 pages.
DOI: 10.1145/3267955.3269023

1 INTRODUCTION

Named Data Networking (NDN) [3] is a proposed data-centric Internet architecture. In an NDN network, communication is accomplished by requesting named and secured data packets. To successfully deploy an architectural change of this scale, it is essential that NDN-enabled nodes are able to communicate over the existing network infrastructure. In pursuing this goal over the last few years, we identified several issues as explained below.

First, in principle, NDN nodes on the same subnet should be able to communicate directly using layer-2 frames. Unfortunately, as we reported in [2], three major roadblocks makes it infeasible in many practical settings: the absence of a standard cross-platform API to use network interfaces, the language restrictions imposed by various platforms, and most importantly, the lack of access to low-level network APIs by the common platforms (e.g. Linux, macOS, and Android), perhaps due to (perceived/potential) security concerns. [1] In such cases, NDN nodes must resort to using IP connectivity for interconnectivity, which can be achieved through UDP/IP tunnels.

Second, setting up tunnels in between requires NDN nodes to discover each other. Again in principle, NDN nodes on the same subnet should be able to discover each other via IP multicast; however in practice, many networks restrict the use of multicast. For example, airport WiFi networks usually disable local multicast support due to the concern of potential abuse by DoS attackers. One may resort to manual configurations to set up tunnels between NDN nodes. However such manual configuration is tedious, error-prone, and difficult to maintain due to various changes over time (e.g. network reconfigurations, new NDN node additions).

Third, in many places network operators even forbid unicast traffic between IP nodes on the same subnet, again due to security concerns of virus and other malware infection.

To summarize: in many cases NDN nodes need to communicate over IP connectivity, i.e. over UDP/IP tunnels, and establishment of these tunnels require the support for automatic discovery of the NDN nodes' IP addresses and their data name prefixes. In case nodes cannot send IP unicast packet to each other, NDN traffic between them may need to travel through an NDN relay node.

In this poster, we describe the design of a rendezvous service for NDN nodes on the same IP subnet to automatically discover each other's IP addresses and data name prefixes, so that they can establish NDN connectivity among themselves by setting up UDP/IP tunnels. We dub this design NDND, NDN Neighbor Discovery.

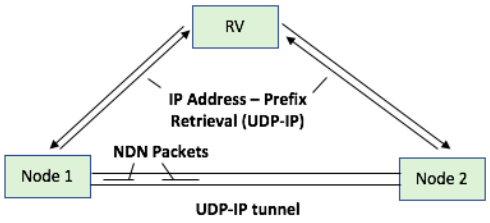

Figure 1: NDN Neighbor Discovery Protocol

The basic idea of NDND is straightforward. We set up a rendezvous server with a well known DNS name N_{RV} and a specific UDP port number, and configure each isolated NDN node with N_{RV} and the port number. [2] An NDN node can then use the standard DNS resolution to learn N_{RV}'s IP address, sending its own IP address and NDN data names in a UDP packet to N_{RV}, and receiving

ICN '18, Boston, MA, USA
 978-1-4503-5959-7/18/09…$15.00
DOI: 10.1145/3267955.3269023

[1] We note that single-app platforms, such as RIOT-OS and Electric Imp, usually give the application full access to the system, including direct low-level APIs.
[2] We configure NDN nodes with N_{RV}'s DNS name instead of directly using IP addresses for agility, sheltering NDN nodes from the number of N_{RV}s and their address changes.

the information about other NDN nodes from N_{RV}. When IP unicast between certain NDN nodes is also prohibited, N_{RV} can also help forward NDN Interest and Data packets between these nodes.

The discussion in this poster focuses on the simplest case of interconnecting isolated NDN nodes on the same subnet. We plan to expand the NDND service to interconnections of these islands across wider areas via similar tunneling over IP.

2 NEIGHBOR DISCOVERY PROTOCOL

The NDND service is made of three pieces: rendezvous server(s) RV, a neighbor discovery application (nd-app) running at each NDN node, and a Neighbor Discovery Protocol. The nd-app collects the local node's name prefixes, and uses the Neighbor Discovery protocol to communicate with the rendezvous server RV to report its own IP address and the prefixes it serves, and retrieve the information about other NDN nodes. We start with the assumption that the local network prohibits IP multicast but allows IP unicast between hosts on the same network. We also assume that all the involved NDN nodes and the RV share a common trust anchor, so that they can authenticate each other's data exchanges.

2.1 Collecting application prefixes on each node

Each application running on an NDN node registers the prefixes it serves with the nd-app by sending an Interest with name "`/ndn/servicediscovery/prefixregistration`", containing the prefixes in the Interest's *Parameters* field [1]. For example, a camera app may register the prefix "`/username/camera`". The nd-app at the node collects all the prefixes in preparation for sending them to RV.

2.2 Communication between node and RV

Each node's nd-app must be configured with an RV name, which it can resolve to an IP address using DNS. The nd-app then communicates with RV using the NDND protocol whose packets are carried in UDP messages, as shown in Figure 2.

A node N's nd-app aggregates its prefixes and IP address into a message of type "`IP_PREFIX_MAPPING`", uses its key to sign it, and sends the message to RV. If N is behind a NAT router, the IP address it sends should be its private address, for communication with other nodes in the same local network. When RV receives N's "`IP_PREFIX_MAPPING`" message, it first authenticates the message, then adds the mapping to its local storage of all the IP-Prefix mappings it has received. If an entry for the IP address exists already, RV replaces the prefix list with the newly received list. Otherwise, RV creates a new mapping. RV then responds to N with a message of type "`IP_PREFIX_MAPPING_LIST`", which contains all known mappings. If N does not receive a response within some expected time period, it will retransmit the "`IP_PREFIX_MAPPING`" message. To keep itself up to date, nd-app sends an "`UPDATE_REQUEST`" periodically. In response, the RV sends its latest "`IP_PREFIX_MAPPING_LIST`".

2.3 Storage and usage of information

Once a node receives IP-prefix mappings for other nodes, it can create UDP-IP tunnels to them and store in its FIB the prefixes that can be served by each tunnel. However, it is sometimes wasteful

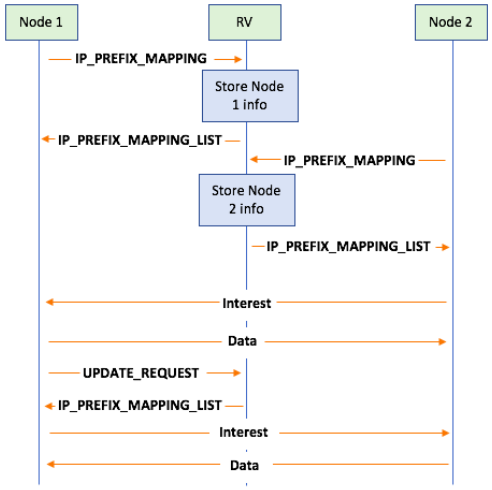

Figure 2: Neighbor Discovery Protocol

for every node to store every prefix it receives. For example, in a building maintenance application, sensors need to communicate with the repo but not with other sensors. Therefore, individual applications within a node must explicitly request to the nd-app their prefixes of interest. The form of this request is a standard NDN Interest packet starting with "`/ndn/servicediscovery/prefixrequest`". The nd-app will filter the IP-Prefix mapping list received from RV and only add routes for requested prefixes. This helps to maintain only relevant information in the FIB.

3 FUTURE WORK

Our current protocol serves nodes in local networks that restrict multicast but allow unicast. As a next step, we plan to extend Neighbor Discovery to support nodes in networks that also restrict unicast, which is common in public WiFi networks. In such networks, the rendezvous server needs to relay all NDN communication between NDN nodes. Another extension is to address interconnection of nodes beyond a local network. We also plan to adjust the protocol to allow optimization at RV before it sends its mapping list back to each node. Finally, while the current communication between each node and RV uses UDP messages, there is benefit in changing this communication to NDN Interest and Data exchange, to make it independent from underlying transport method (only the tunnel establishment part needs to change, which is straight-forward).

ACKNOWLEDGMENTS

This work is partially supported by the US National Science Foundation under awards CNS-1719403, CNS-1629922, and CNS-1629769.

REFERENCES

[1] NDN Packet Format Specification. https://named-data.net/doc/ndn-tlv/.
[2] Wentao Shang, Alex Afanasyev, Yanbiao Li, Jeff Burke, and Lixia Zhang. Device-to-device communication with Named Data Networking. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017.
[3] Lixia Zhang et al. Named Data Networking. *ACM Computer Communication Review*, July 2014.