Integrating IP and Future Networks through a Performance Enhancing Proxy

Kristjon Ciko
University of Oslo, Norway
kristjoc@ifi.uio.no

***Abstract*—Recent research has been focusing on exploring new network architectures to address the fundamental issues of the current Internet. While plenty of new architectures have emerged over the years, their deployment is facing many challenges. To ease the integration and experimentation of such architectures, and thus potentially advance their gradual deployment, we introduce a proxy called "Performance Enhancing Proxy for Deploying Network Architectures (PEP-DNA)". By "translating" between TCP/IP and the new architecture, PEP-DNA enables TCP/IP-based applications to gain the benefits of future technologies without having to be redeveloped. We consider two novel architectures to illustrate our proxy usage, the Recursive InterNetwork Architecture (RINA) and Information-Centric Networking (ICN), but PEP-DNA can be extended to support other architectures. In this demo we walk through realistic scenarios of using PEP-DNA and cover the necessary steps needed for enabling interoperability between TCP/IP applications and a RINA or an ICN network.**

***Index Terms*—Future Internet, Network Architectures, Performance Enhancing Proxies**

I. Introduction

The number of users and applications in the Internet has grown rapidly since the Internet was first launched. With the significant increase of the data traffic demands, the current Internet architecture is facing major issues related to security, multi-homing, mobility and Quality of Service (QoS). Several extensions have been added into the original TCP/IP networking stack to address each issue individually. On the other hand, many researchers have followed a different approach by proposing new network architectures which attempt to solve these issues [1]–[5]. Nevertheless, there is not much evidence of the usage of new networking models in real-life networks. The current Internet architecture is found to be difficult to evolve and new technologies require careful downward-compatibility considerations to survive and co-exist [6]–[8].

When all hosts and network devices along an end-to-end path support a new architecture, it is possible to directly "switch over" and benefit from the new communication model. The authors of [9] have shown that this can be done with minimal performance overhead, using RINA [4]. To address the deployment problem of new architectures, McCauley et al. designed an architectural framework called "Trotsky", which provides mechanisms for tunneling traffic over legacy Internet domains [10]. However, such tunneling can face several challenges, for example, the "new" traffic can cause harm to other Internet traffic, and the endpoints need to be upgraded.

As an alternative to tunneling, in this paper we focus on the approach of *translating* between the Internet and an alternative network architecture. We introduce the design and implementation of PEP-DNA, a TCP proxy which enables TCP/IP applications to send traffic over a different architecture and vice versa, without having to be redeveloped. PEP-DNA falls into the category of "split connection" Performance Enhancing Proxies (PEPs), which are described in detail in RFC3135 [11] (we will also briefly introduce this design in the next section). Our proxy is a kernel-based implementation which aims to achieve high throughput with low latency[1] and can be installed wherever a translation from one technology to another needs to occur in the network.

PEP-DNA is currently able to interconnect a TCP connection with (i) another TCP connection, (ii) the RINA architecture, and (iii) an ICN network. However, because of its modular and flexible design, PEP-DNA can be extended to support other technologies with minimal changes. We chose RINA and ICN as case studies architectures, considering the potential benefits their deployment could bring in networking. RINA is secured by design [13], and it can natively provide mobility and multi-homing. Moreover, since RINA is a recursive and scope-aware architecture, it can naturally support in-network congestion control [14]. ICN is another promising architecture for future networks, which is very different from both RINA and TCP/IP. ICN is built on the concept of naming information/content/data rather than hosts. Unlike IP-based networks, ICN endpoints share named content, leading to more flexible, scalable and secure networks [15].

The rest of the paper is structured as follows: in Section II we provide implementation details and introduce the main components of PEP-DNA. We describe the demo scenarios and how we interconnect different domains through our proxy in Section III. Finally, in Section IV we conclude the paper and provide additional information needed to run the demo experiments.

The authors were part-funded by the Research Council of Norway under its "Toppforsk" programme through the "OCARINA" project (https://www.mn.uio.no/ifi/english/research/projects/ocarina/).

[1]Due to the lack of space, we do not present performance metrics in this paper, but only perform a simple performance evaluation of our proxy during the demonstration. An extended version of the paper can be found in [12].

978-1-6654-2434-9/21/$31.00 ©2021 IEEE

II. PEP-DNA Design and Implementation

We implement PEP-DNA as a Linux kernel module that can be loaded in recent kernel versions. It does not require any hardware modification nor changes of the network stack, thus making it easy to deploy in existing networks. It runs entirely in the kernel space, next to the TCP/IP stack. While developing tools in the kernel increases the level of complexity, this design provides better performance for high-speed networking [16]. In-kernel solutions improve the overall system performance by reducing the context switching overhead added by data exchange between the kernel and user space.

PEP-DNA is classified as a connection-splitting proxy: it terminates the connection from the sender and establishes a new connection to the receiver or to the next PEP within the new network architecture. By spoofing the IP addresses and TCP ports, our proxy operates in a fully transparent way. The end hosts are not aware of the connection splitting performed by the proxy. Fig. 1 shows a sequence diagram describing the exchange of messages and data during the connection establishment and data transfer phases of an inter-domain communication between two TCP applications over a different architecture.

To achieve the behaviour shown in Fig. 1, PEP-DNA integrates three main components: (i) an IP-layer Netfilter[2] hook, (ii) the connection handler (a.k.a the connector), and (iii) the data relay engine. The Netfilter hook enables the proxy to be transparent by intercepting every incoming SYN packet and storing its 4-tuple information (source IP address, source port, destination IP address, destination port) in a hash table. This information is used by the connector of the ingress proxy to initiate a new connection to an egress proxy inside another network architecture cloud. After all split connections are established, the data relay engine forwards the traffic from one technology to another. In the next section, we describe more in detail the main components of the proxy using two real-life scenarios as part of the demonstration content.

III. Demo Scenarios

The local testbed that we use for this demo consists of five virtual machines, connected as in Fig. 2 (we replace the three interior routers of the new architectures with one, for the sake of simplicity). All machines run Debian 10 operating system with kernel version 4.19. We choose RINA and CCN (CCN; Content-Centric Networking is a prominent ICN architecture. We use CCN and ICN interchangeably in this paper.) as examples of novel network architectures to connect to. At the proxy nodes and the router in between, we install the CCN-lite[3] implementation of CCN and the IRATI implementation of RINA [17] as a protocol stack alongside TCP/IP.

The demo is comprised of two scenarios. In the first scenario (TCP-RINA-TCP), we show how using our proxy can enable TCP applications to operate seamlessly through a RINA domain. We plan to connect a couple of well-known TCP applications which follow the client-server model, such as *iperf*, *cURL* and *Nginx*. We also plan to evaluate the performance of the proxy using *iperf*, and compare with the case when the TCP applications talk to each other via a full TCP/IP path. In the second scenario (TCP-CCN), we show the integration of a TCP HTTP client into a CCN network using PEP-DNA.

Fig. 2 illustrates the necessary steps for establishing a communication channel and sending data from the TCP client to the server. In Step 1 of Fig. 2, the client initiates a connection to the server by sending a SYN packet with destination IP address 8.4.4.1 and destination port 80 (the IP address and the TCP port on which the server is listening). After the Netfilter hook of the Ingress PEP intercepts the SYN packet and stores its 4-tuple (source IP address, source port, destination IP address, destination port) in a hash table, the SYN packet is redirected unchanged by TPROXY iptables[2] to a local socket of the proxy. The connector immediately initiates a connection establishment procedure within the new architecture (RINA, in this case), illustrated in Step 2. PEP-DNA leverages the Naming and Addressing principles of RINA, which identify an application by its Application Process Name (APN) and Application Process Instance (API). For every incoming TCP connection request, the proxy spawns a new API to send a RINA request for allocating a flow towards the other PEP-DNA (Egress PEP in Fig. 2). The end-point IP addresses and TCP ports are concatenated as a string which represents the name of the API.

In Step 3 the Egress PEP's connector extracts the end-point information from the requesting API given in the flow request header, initiates a TCP connection to the destination IP address and port of the server, and sends back a positive flow response to the Ingress PEP only after the Egress PEP - TCP server connection is established. After receiving the flow response, the Egress PEP sends a SYN/ACK packet to the client. In this way the proxy connects with both the client and the server as early as possible and also achieves basic *fate-*

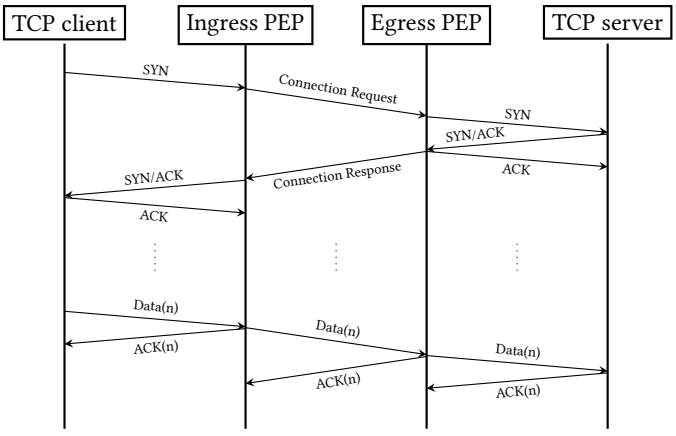

Fig. 1: Connection Establishment and Data Transfer phases of an inter-domain communication scenario where two PEP-DNA proxies enable two TCP applications to operate over a new network architecture.

[2]https://netfilter.org

[3]https://github.com/cn-uofbasel/ccn-lite

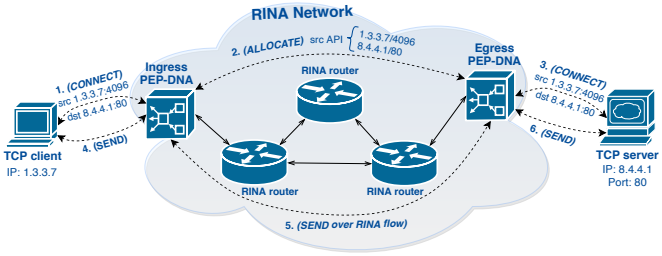

Fig. 2: Inter-domain communication between two TCP applications through proxies deployed at the RINA-IP borders.

sharing at the same time (the client is not made to believe that it has a connection before the connection to the server really is established).

Once the connection between the client and server is established, the proxies take care of forwarding data directly in the kernel, avoiding context switching overhead and allowing zero-copy transmission. Both the client and the server are not aware of the connection splitting performed by the proxies, since PEP-DNA operates in a transparent way.

Fig. 3 shows the second scenario (TCP-CCN) of the demo. In this scenario, we use only one proxy to integrate TCP applications with a CCN network. After establishing a TCP connection with the proxy, the HTTP client requests through the HTTP protocol a content which is identified in the CCN network by the hierarchical name *"/ndn/test/content"*. PEP-DNA parses the HTTP GET request, constructs a CCN request for content, with the interest being *"/ndn/test/content"*, and injects it in the CCN network. Then, the interest request is propagated inside the CCN network until it reaches the CCN Web server where the content is stored. In reply to the interest request, the CCN Web server sends the content back to the request originator (PEP-DNA), which forwards it to the TCP HTTP client.

IV. Conclusions

In this paper we have presented the design and implementation of PEP-DNA, a TCP connection-splitting proxy that can interconnect TCP with another TCP connection or with an entirely different network architecture. We show through two realistic scenarios how the proxy is able to efficiently interconnect TCP/IP applications with a RINA or an ICN network and vice versa. Due to its modular design, PEP-DNA can be easily extended to support other new technologies, and thus potentially advancing the gradual deployment of new network architectures. An extended version of the paper where we thoroughly evaluate the performance of our proxy can be found in [12]. The PEP-DNA source code as well as the instructions needed to replicate this demo are published in a public repository[4].

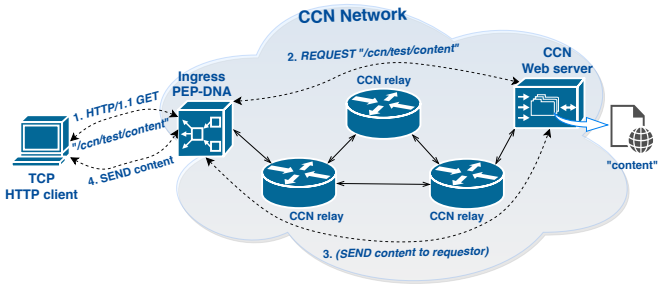

Fig. 3: Interconnection between a TCP HTTP client and a CCN network.

References

[1] A. Afanasyev, J. Burke, T. Refaei, L. Wang, B. Zhang, and L. Zhang, "A brief introduction to named data networking," in *IEEE MILCOM*, 2018, pp. 1–6.

[2] C. Severance, "Van Jacobson: Content-centric networking," *Computer*, vol. 46, no. 1, pp. 11–13, 2013.

[3] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *SIGCOMM*. New York, NY, USA: ACM, 2007.

[4] J. Day, *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall, 2008.

[5] J. Pan, S. Paul, and R. Jain, "A survey of the research on future internet architectures," *IEEE Communications Magazine*, vol. 49, no. 7, 2011.

[6] M. Handley, "Why the internet only just works," *BT Technology Journal*, vol. 24, no. 3, pp. 119–129, Jul. 2006. [Online]. Available: https://doi.org/10.1007/s10550-006-0084-z

[7] G. Papastergiou, G. Fairhurst, D. Ros, A. Brunstrom, K. J. Grinnemo, P. Hurtig, N. Khademi, M. Tuexen, M. Welzl, D. Damjanovic, and S. Mangiante, "De-ossifying the internet transport layer: A survey and future perspectives," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 619–639, Firstquarter 2017.

[8] H. Balakrishnan, S. Banerjee, I. Cidon, D. Culler, D. Estrin, E. Katz-Bassett, A. Krishnamurthy, M. McCauley, N. McKeown, A. Panda, S. Ratnasamy, J. Rexford, M. Schapira, S. Shenker, I. Stoica, D. Tennenhouse, A. Vahdat, and E. Zegura, "Revitalizing the public internet by making it extensible," *SIGCOMM Comput. Commun. Rev.*, vol. 51, no. 2, p. 18–24, May 2021. [Online]. Available: https://doi.org/10.1145/3464994.3464998

[9] K. Ciko and M. Welzl, "First contact: Can switching to RINA save the internet?" in *IEEE ICIN*, 2019, pp. 37–42.

[10] J. McCauley, Y. Harchol, A. Panda, B. Raghavan, and S. Shenker, "Enabling a permanent revolution in internet architecture," in *SIGCOMM*. New York, NY, USA: ACM, 2019, pp. 1–14.

[11] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135 (Informational), RFC Editor, Fremont, CA, USA, Jun. 2001. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3135.txt

[12] K. Ciko, M. Welzl, and P. Teymoori, "PEP-DNA: a performance enhancing proxy for deploying network architectures," in *2nd Workshop on New Internetworking Protocols, Architecture and Algorithms 2021 (NIPAA-21)*, Nov. 2021.

[13] G. Boddapati, J. Day, I. Matta, and L. Chitkushev, "Assessing the security of a clean-slate internet architecture," in *IEEE ICNP*, Oct 2012.

[14] M. Welzl, P. Teymoori, S. Gjessing, and S. Islam, "Follow the model: How recursive networking can solve the internet's congestion control problems," in *IEEE ICNC*, 2020, pp. 518–524.

[15] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1–12.

[16] Y. Lin, C. Ku, Y. Lai, and C. Hung, "In-kernel relay for scalable one-to-many streaming," *IEEE MultiMedia*, vol. 20, no. 1, pp. 69–79, 2013.

[17] S. Vrijders, D. Staessens, D. Colle, F. Salvestrini, E. Grasa, M. Tarzan, and L. Bergesio, "Prototyping the recursive internet architecture: the IRATI project approach," *IEEE Network*, March 2014.

[4] https://github.com/kr1stj0n/pep-dna.git