Efficient Transport Layer and Socket API for ICN

Mauro Sardara
Cisco Systems Inc.
msardara@cisco.com

Luca Muscariello
Cisco Systems Inc.
lumuscar@cisco.com

Alberto Compagno
Cisco Systems Inc.
acompagn@cisco.com

ABSTRACT

In this demonstration, we showcase a transport layer and socket API [10] that can be used in several ICN architectures such as NDN, CCN and hICN [8]. The current design follows the successful BSD socket approach: a simple API that can be easily inserted in current applications and used to develop novel ones. In the PoC, we compare the performance of some of the transport services provided by both our transport layer and the today's transport layer: reliable communication, data segmentation and reassembly, data integrity. Moreover, we show the benefits of adopting our transport layer in existing application in terms of CPU load reduction and a lower memory consumption.

CCS CONCEPTS

• **Networks** → **Programming interfaces**; Network experimentation;

KEYWORDS

ICN, Transport Services, Socket API

ACM Reference Format:
Mauro Sardara, Luca Muscariello, and Alberto Compagno. 2018. Efficient Transport Layer and Socket API for ICN. In *5th ACM Conference on Information-Centric Networking (ICN '18), September 21–23, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3267955.3269013

1 INTRODUCTION

Since the release of BSD Socket API, internet applications rely on transport services to inter-operate with the network and move data across the Internet. The key to success of such API is its simplicity: developers can send/receive data to/from the network as if they were programming sequential-access files applications. Moreover, by setting the corresponding socket options, developers can exploit the required transport service (e.g., reliable/unreliable communication, flow control, congestion avoidance, data segmentation and reassembly, etc) implemented by a transport protocol that hides the complexity to the application.

Information Centric Networking (ICN) is a relatively novel network architecture that enables a simplified and more efficient user-to-content communication. While a considerable amount of work has been done in designing and developing the network layer with several proposed architectures (NDN, CICN, hICN), a smaller effort has been done in designing and evaluating the transport layer. Most of the work on transport layer for ICN has focused on receiver-driver congestion control protocols (surveyed in [9]) but only few target the design of a socket API [4, 7]. Both of [4, 7] define a new communication abstraction model based on the consumer/producer principle of ICN. While the work in [4] adapts the BSD Socket API definition to the consumer/producer paradigm, the work in [7] defines a new set of API, specific to ICN. Unfortunately, none of them provide an evaluation of their implementation and a comparison with today's transport layer.

In this demonstration, we present a transport layer with an API reflecting the same simplicity of the BSD socket API, that provides a wide set of transport services: data-oriented communication, data segmentation and reassembly, traffic flow, congestion control, synchronous and asynchronous data publication, data integrity and origin authentication. Considering the video distribution use case, we show the performance of our implementation, able to reach an application goodput of 5.79Gbps, we compare our reliable transport protocol with TCP and finally we show the benefits brought by our transport at application layer.

2 TRANSPORT SERVICES

The services and the API offered by the transport layer to applications depend on their communication model: the content producer publishes data under a given name, by segmenting, naming and authenticating them with its identity. On the other hand the consumer fetches the data packets, optionally using a flow control algorithm, verifies their origin and reassembles them. Being the operations of consumers and producers disjointed, we identify two kind of communication sockets, each with its specific transport services and API.

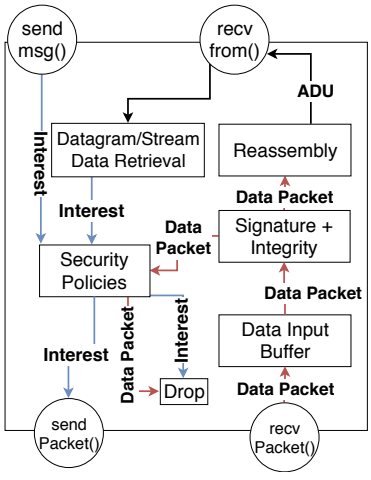

Figure 1: Consumer socket

ICN '18, September 21–23, 2018, Boston, MA, USA

ACM ISBN 978-1-4503-5959-7/18/09.
https://doi.org/10.1145/3267955.3269013

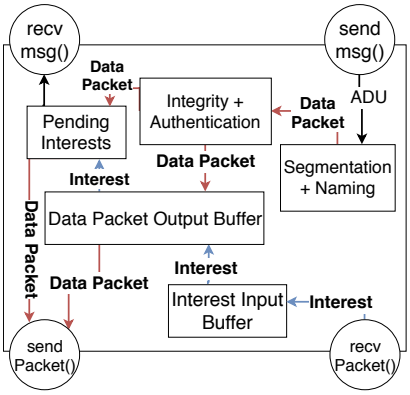

Figure 2: Producer socket

The *consumer socket* is in charge of retrieving application data from multiple potential producers whereas the *producer socket* is in charge of publishing application data units (ADUs) for multiple potential consumers. This design allows applications to exchange data in a multi-point to multi-point manner, overcoming the limitations of connection-oriented transports such as TCP.

In this demonstration, the transport services showcased are (1) *synchronous ADU segmentation and naming*, (2) *ADU fetching and reassembly*, (3) *congestion and flow control*, (4) *integrity, authentication and data origin verification*. Fig. 1 and 2 show how services operate inside the two sockets.

Transport services are exposed to applications using the socket interface extensions for IPv6 [5]: application data units can be published using the `sendmsg` system call with a producer socket and can be retrieved using the `recvfrom` system call with a consumer socket.

3 DEMONSTRATION

The demonstration can be decomposed in three parts: (1) we show how to write new and integrate existing applications using our socket API; then (2) we compare our implementation with TCP, underlying if and how ICN specific services like authentication and integrity affect the communication; finally (3) we show what benefits an ICN transport can bring to the application. For the first two parts we use a simple topology with two hosts (consumer/producer in ICN, server/client in IP/TCP), connected through a 10Gb/s link. In the third part, we deploy a more complex topology in which a group of video clients is connected to an Apache Traffic Server (ATS) [1] through a 10Gb/s link. ATS is configured as an HTTP reverse proxy and a Web cache. As origin server we use an nginx [2] server live-feeded by a RTMP stream generated by the Open Broadcaster Software (OBS) [3]. We stream 48 channels, with 2 second segments. To compare our ICN transport with TCP, we connect the ATS plugin to our ICN sockets and we create one producer socket per channel. In all the three parts, we use our Hybrid ICN (hICN) implementation that brings NDN/CCN into IPv6 [8].

The demonstration showcases the following features of the ICN transport:

1) **Easiness of integration inside applications**: The usage of an API as simple as the BSD socket API aims to facilitate the integration of ICN transport services into applications and allows to decouple the application logic from the specific ICN implementation (NDN, CICN, hICN).
2) **High speed userspace implementation**: We demonstrate how our ICN transport, based on VPP [6] and DPDK [11], is able to operate at high speed exploiting zero-copy and userspace packet processing.
3) **Integrity and Authentication as a transport built-in**: Signing and verifying packets have a significant impact on the overall transport performance, both in terms of application goodput and latency. This overhead can be reduced by offloading them to dedicated hardware. The PoC aims to show how much a secure transport can affect the communication between consumers and producers.
4) **Multicast and Server Load Reduction**: A fundamental difference between a TCP and an ICN socket is the fact that a ICN socket can serve/retrieve data to/from multiple destinations/sources. This allows to significantly reduce the server load, which can produce data once, instead of sending it to each client in unicast. Therefore, consumers' requests can be serviced by the transport itself once the data is made available after production.

4 CONCLUSION

Through this demonstration, we illustrate what benefits an ICN transport layer can provide to today's applications and developers, as well as how the basic security features of ICN affect the communication among the network entities in terms of goodput and latency.

While the socket API allows easy extension and integration inside today's applications and the software implementation features high speed packet processing, the cryptographic operations still have a significant impact on the transport performance. Our software implementation is still feasible for a large set of applications, but hardware acceleration is required for a real ICN deployment.

REFERENCES

[1] Apache Traffic Server. http://trafficserver.apache.org/.
[2] *nginx*. https://nginx.org/en/.
[3] Open Broadcaster Software (OBS). https://obsproject.com/.
[4] M. Gallo et al. 2014. NaNET: Socket API and Protocol Stack for Process-to-content Network Communication. In *Proc. of the 1st ACM SIGCOMM ICN Conference (ACM ICN'14)*.
[5] R.E. Gilligan et al. 2003. *Basic Socket Interface Extensions for IPv6*. Technical Report 3493. https://rfc-editor.org/rfc/rfc3493.txt
[6] Linux Foundation FD.io. Vector Packet Processing. https://fd.io.
[7] I. Moiseenko et al. 2015. Consumer / Producer Communication with Application Level Framing in Named Data Networking. In *Proc. of the 2nd ACM SIGCOMM ICN Conference (ACM ICN'15)*.
[8] L. Muscariello et al. 2018. *Hybrid Information-Centric Networking*. Internet-Draft draft-muscariello-intarea-hicn-00. Internet Engineering Task Force. Work in Progress.
[9] Y. Ren et al. 2016. Congestion Control in Named Data Networking - A Survey. *Comput. Commun.* 86, C (Jul 2016), 1–11.
[10] M. Sardara et al. 2018. A Transport Layer and Socket API for (h)ICN: Design, Implementation and Performance Analysis. In *Proceedings of the 5th ACM Conference on Information-Centric Networking (ACM ICN'18)*.
[11] The Linux Foundation. Data Plane Development Kit. https://dpdk.org.