Poster: User-space Networking Libraries & Control Plane Negotiations for Seamless Multi-connectivity

Seppo Hätönen
University of Helsinki

Ashwin Rao
University of Helsinki

Sasu Tarkoma
University of Helsinki

***Abstract*—Offering seamless connectivity to devices capable of simultaneously using multiple communication interfaces continues to be a hard problem. This problem is important for edge computing because edge services may be available only on a subset of networks to which the device is capable of connecting to. We argue that various aspects of this problem can be addressed by leveraging the current trends of using user space libraries for networking, and allowing control plane negotiations between user devices and networks.**

***Index Terms*—Multi-connectivity, Control Plane, user space networking libraries, Edge Computing.**

I. Introduction

Devices with multiple communication interfaces can be simultaneously connected to one or more networks. These networks may offer different capabilities and edge services to the device. For instance, the cellular network can offer video streaming services and the Wi-Fi network can offer access to local edge computing services. However, typically the devices only allow the usage of a single network for data transfer, and usually the Wi-Fi has higher priority than cellular due to different reasons such as monetary, bandwidth or latency [1]. In this context, managing multi-connectivity across different networks continues to be hard. Specifically, *the device needs to decide which interfaces can be used by flows of an edge services when multiple interfaces offer connectivity, and this decision must not break connectivity to other networks and services used by the device.*

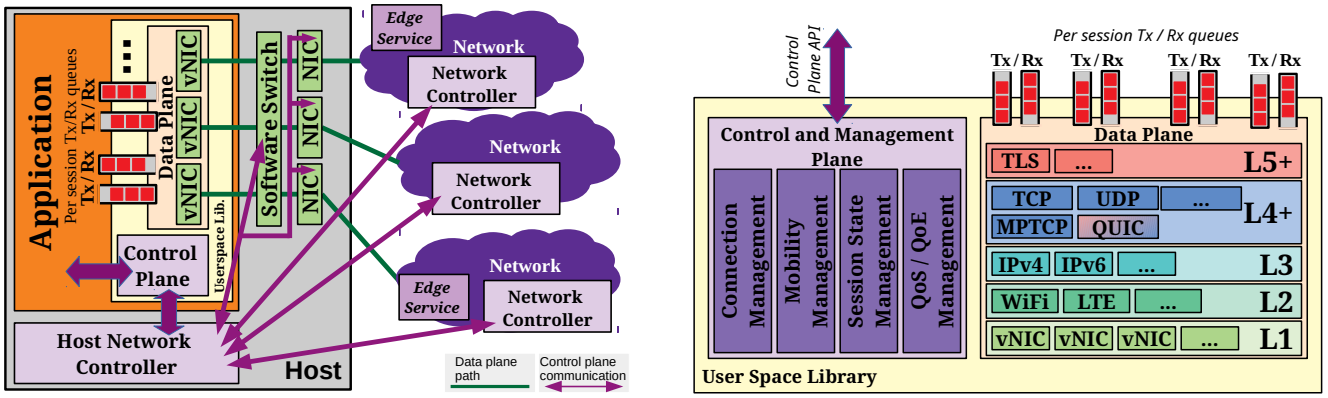
Offering seamless connectivity between networks is not trivial, and it contains multiple facets that need to be addressed. For example, different networks use different IP address spaces, a network might offer connectivity only via a proxy, or the middleboxes in the network may modify the packet headers [2]. Similarly, edge computing services are typically only available in limited number of networks. For instance, edge services hosted on Amazon Wavelength are available only on specific operator networks [3]. The problem of offering seamless multi-connectivity therefore requires addressing sub-problems that span multiple layers of the protocol stack, while also requiring inputs from the networks to which the device is connected.

The key sub-problems of this problem include i) seamless connection establishment and mobility in available networks, ii) support from multiple protocols, and iii) support for control plane negotiations. From the user perspective the connection establishment can be either seamless, i.e. the networking stack used by the application takes care of connection establishment, or it can be transparent, i.e. applications have to explicitly establish new connections. For instance, Multipath TCP (MPTCP) offers seamless connection establishment and mobility for TCP [4], while the approach of Hätönen *et al.* [5] offers seamless mobility only within a network but is not limited to any particular protocol. Connection establishment is protocol specific, and applications are increasingly designed to be able to exchange data using a variety of protocols. For instance, an edge service may support MPTCP and QUIC [6] as possible transport protocols. In this context, the solution must also determine which of the supported protocols should be used when communicating with the available services. This requires control plane negotiations with the network to determine protocols the network and the edge services support.

As shown in Figure 1, we argue that the problem of offering seamless connectivity can be addressed by leveraging the current trends of using user space libraries for networking, and allowing control plane negotiations between user devices and networks and services. As detailed in §II, decoupling control and data plane using user space libraries, allows the host to offer seamless connection establishment and mobility, support multiple protocols, and support control plane negotiations. Our work builds on the insights of the URLSession library [7] which exemplifies some of the benefits of leveraging user space libraries for networking.

II. System Architecture

Overview. As shown in Figure 1(a), our system consists of two components on the host machine: the host network controller, and a user space library. Our library decouples the control and data plane for data exchange making our approach drastically different from the traditional sockets interface. It creates a Tx/Rx queue for each data exchange session involving the application; the application uses these queues instead of sockets to exchange data. At the same time, the application can use the control plane API to configure the data exchange and specify its requirements from the network. For instance, applications can use this API to specify a) the interfaces that should be used: prefer Wi-Fi but do not break connections over cellular, *etc.*; b) the details for the connection establishment including the hostname, the supported transport layer protocols, the security credentials, *etc.*, c) the Quality-of-Experience (QoE) or Quality-of-Service (QoS) requirements for the data exchange, and d) the services to be used. The host network controller manages the networking interfaces of the host machine, and this module is also responsible



(a) **System Components.** *The user space library decouples the control and data plane of data exchange sessions. It exposes a Tx/Rx queue for each session, and it contains a virtual NIC for each host NIC. All NICs including the host NICs are connected to a software switch managed by the host network controller. The host controller aids the control plane of the library offer seamless connectivity. It also proxies the QoS/QoE requests made to the control plane of the application library and networks offering connectivity to the host.*

(b) **User space library for seamless connectivity.** *The library will consists of modules for the control plane and data plane. The control plane modules are responsible for managing the seamless connectivity and QoS/QoE negotiations. The data plane modules are responsible for exchanging data using the protocols specified by the control plane. Instead of using sockets, the applications use per-flow Rx/Tx queues for the data exchange, and a dedicated control plane API to exchange the control plane information.*

Fig. 1. **System Design.** *Our system aims to offer seamless connectivity by leveraging recent advances in user space libraries for networking and the ability to negotiate the application requirements and demands with the networks offering connectivity and edge services.*

for negotiating the requirements of the applications with the controllers of the networks offering connectivity to the host.

Control Plane. Along with a dedicated network controller in the host machine, we assume that each network offering connectivity to the host has its own controller. As shown in Figure 1(a), the host controller uses a dedicated control plane channel for exchanging the control plane information with the network controllers. For instance, the controller can use the MAMS protocol [8], or the work of Rezende *et al.* [9]. Furthermore, this channel can also be used by the controllers in the network to inform hosts about network capabilities and services such as the edge services. As shown in Figure 1(a), the host controller uses the control plane API to exchange the control plane information with the user space library. Specifically, it aids the various control plane activities of the library such as connection management, mobility management, and managing the session state. For instance, it aids connection management by notifying the library when a new interface is available for the data exchange.

Data plane module in the user space library. As shown in Figure 1(b), the data plane module is responsible for performing the data exchange with the remote hosts. It exposes a Tx/Rx queue for exchanging data for a given session, and it uses the protocols specified by the application. For instance, an application may specify that it supports data exchange only using TLS, MPTCP, and IPv6. Furthermore, it performs this data exchange using the interfaces made available to the library by the host network controller and authorized by the control plane module. For instance, the host network controller may provide connectivity using multiple interfaces, but the control plane module may be configured by the application to prefer a subset of the interfaces.

Example connection to multiple edge services. Typical mobile devices have access to both Wi-Fi and cellular networks. For example, an edge service A may be available only in the Wi-Fi network, while another service B may be available only in the cellular network. Using service B would normally require disconnecting from Wi-Fi and switching to cellular network, thus breaking the access to service A.

The application begins by using the control plane API to query for available edge services in the connected networks. The application then specifies that it wants to exchange data with an edge service preferably using Quic or TCP, IPv6, and any of the possible communication interfaces.

The library takes care of performing the connection establishment and provides a Tx/Rx queue for the data exchange. Furthermore, the application can specify the QoE requirements for the edge service which the host controller can exchange with the network controllers. Based on these negotiations, the host controller provides the control plane module of the application with the list of interfaces that can be used for exchanging data. When one of the interfaces of the host machine moves to another network, the host controller updates the lists of interfaces which can be used. If the new network can satisfy the application requirements, the user space library uses it for creating a new connection, and starts using it for the data exchange after the connection is successfully established. Note that during the data exchange sessions, the application continues to exchange data using the Tx/Rx queues.

III. Discussion

Some of the key enablers for this work include networking namespaces which allow creation of isolated virtual interfaces [10], the ability to perform control plane negotiations [8], [9], and the URLsession library [7]. Furthermore, our work on seamless flow migration [5] helped us envision the benefits of using a network controller on the host machines. Some of the key challenges which we plan to address include the design of the control plane interface and quantifying the impact of control plane negotiations on the QoE.

REFERENCES

[1] "Android Help - Connect to Wi-Fi networks on your Android device," https://support.google.com/android/answer/907584, [accessed 2020-08-26].

[2] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is It Still Possible to Extend TCP?" in *Proceedings of IMC*, 2011.

[3] "AWS Wavelength," https://aws.amazon.com/wavelength/, [accessed 2020-08-26].

[4] C. Paasch and O. Bonaventure, "Multipath TCP," *Communications of the ACM*, no. 4, p. 51–57, 2014. [Online]. Available: https://doi.org/10.1145/2578901

[5] S. Hätönen, T. Huque, A. Rao, G. Jourjon, V. Gramoli, and S. Tarkoma, "An SDN Perspective on Multi-connectivity and Seamless Flow Migration," *IEEE Networking Letters*, 2019.

[6] Y. Cui, T. Li, C. Liu, X. Wang, and M. Kühlewind, "Innovating transport with quic: Design approaches and research challenges," *IEEE Internet Computing*, vol. 21, no. 2, pp. 72–76, 2017.

[7] Apple Inc., "URLSession - Apple Developer Documentation," https://developer.apple.com/documentation/foundation/urlsession, [accessed 2020-08-26].

[8] S. Kanugovi, F. Baboescu, J. Zhu, J. Mueller, and S. Seo, "Multiple Access Management Services," IETF Internet-Draft, Tech. Rep. draft-kanugovi-intarea-mams-framework-04, May 2019.

[9] P. Rezende, S. Kianpisheh, R. Glitho, and E. Madeira, "An SDN-Based Framework for Routing Multi-Streams Transport Traffic Over Multipath Networks," in *Proceedings of ICC*, 2019.

[10] P. B. Menage, "Adding Generic Process Containers to the Linux Kernel," in *Proceedings of the Ottawa Linux Symposium*, 2007.