coin
Internet-Draft
Intended status: Informational
Expires: October 29, 2021

R. Abhishek
Tencent

April 27, 2021

A collaborative Edge-Cloud framework for XR applications
draft-abhishek-coin-xr-edge-cloud-00

Abstract

This document discusses a collaborative edge-cloud model and application of network slicing for Extended Reality (XR), including both Augmented Reality (AR) and Virtual Reality (VR), especially with respect to the architectural framework and "QoS" based optimal latency tolerant resource allocation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 29, 2021.

Table of Contents

1. Introduction

To realize the full capacity of Extended Reality (XR), including both Augmented Reality (AR) and Virtual Reality (VR), a high-end hardware device is required. This requirement arises because XR applications are likely to require a huge amount of processing power and storage to give the user the feeling of being in a truly immersive environment. With the increasing number of XR applications, the requirement for the devices' processing capacity has increased. More importantly, these XR applications require real-time video stream processing to recognize specific objects, besides, some AR application requires generation of new video frames [draft-ietf-mops-ar-use-case-00]. Therefore, the current challenges in using XR have been the capacity, energy consumption, and weight of the device. All of these are arising due to the massive processing requirement of the applications running on the device. Heavy devices result in the user having an uncomfortable experience, and high processing capacity makes the device expensive. Besides, with limited resource availability at the device, processing tasks that require more than available resources would add computational and processing latencies. Therefore, there exists a gap between the capabilities of the current state of the art and the requirements for the future.

One way to overcome this is by offloading processing to network-based resources in the edge and the cloud. However, the challenge is to minimize the latency when the processing is offloaded to the upper layers (edge and cloud) [Figure 1]. There are lots of contributing factors to this latency, such as sampling delay, computation delay including image processing and frame rendering delay, networking delay comprising of queuing and transmission delay. Therefore, an optimized architecture is required in order for the computational and communicational delay not to throttle the XR system. This document

talks about splitting a task among the user, edge, and the cloud and applying network slicing to minimize the latency experienced by the user, and provide QoS-based traffic routing resource allocation on the latency bounds for different traffic classes. Besides, using a latency-bound network may result in the user having motion sickness [Latency-Network-AR-VR]. In this regard, using network slicing would work to the user's advantage by dedicating a specific virtual slice for XR, thereby improving the Quality of Experience.

2. Architectural Framework

The concept of using edge computing and cloud computing for offloading the processing from the user device to the upper layers has garnered much interest from the industry and academia in recent years. Edge computing brings real-time data processing near the user, thereby running the application in closer network proximity to the (XR) user devices. Processing the applications as close to the user as possible compared to running them on a centralized cloud or data center helps reduce the transmission latency. With their high computational capabilities, the cloud servers can handle resource-intensive tasks requiring CPU and GPU-like processing. Therefore, using a collaborative model comprising the user (XR), the edge, and the cloud is more optimal for performance and latency reduction.

Network slicing allows partitioning the physical network into logically isolated sub-networks for flexible and optimized resource provisioning. Thereby, one or more network slices can be completely dedicated to the needs of XR. Each slice can host one or more Network Slice Subnet Instance (NSSI) [I-D.draft-defoy-coms-subnet-interconnection-04] for different application needs. These network slices may have slice priority linked to it, which may help in resource allocation during stressed situations [Spartacus]. This slice priority may be helpful in resource allocation based on the traffic class.

The architectural framework is shown in the figure below. It can be partitioned into three layers: the user, the edge, and the cloud layers. The edge and the cloud will have different network slices for the different traffic classes. A task may be divided among the user, edge, and cloud layers. The processing split among the user layer, edge layer, and the cloud layer further adds to optimization and reduced delay. A task module is present at the user and the edge layer to split the task. The user layer's task module decides which tasks are to be processed at the user's device and which tasks to be sent to the upper layers for further processing.

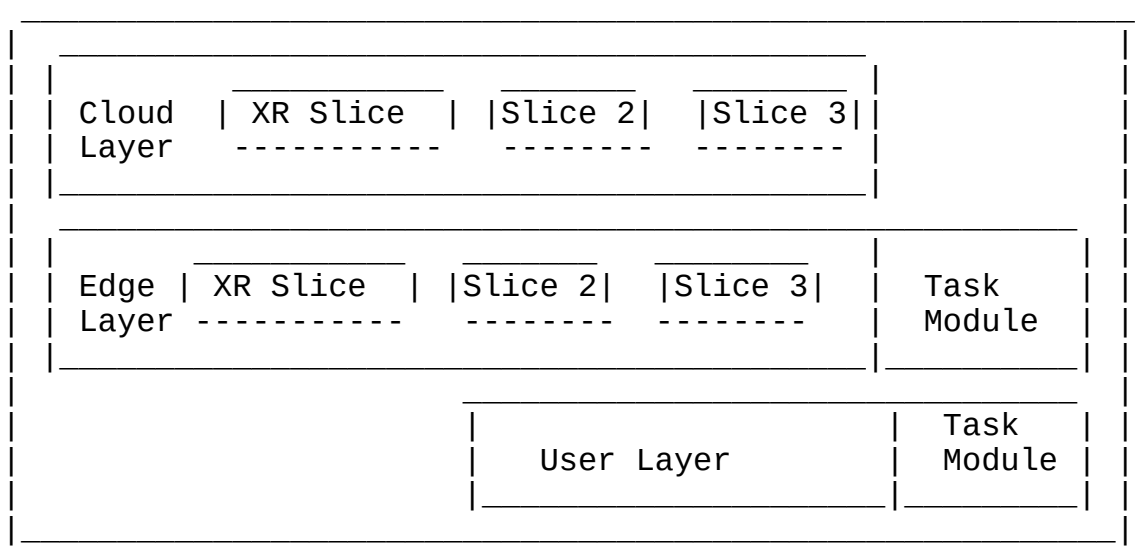

Figure 1: Architectural Framework

2.1. A collaborative Edge-Cloud model

An effective collaboration among user, edge, and cloud layers is important for optimal performance and latency improvement. Processing at the network edge helps overcome cloud offloading shortcomings, such as long latencies and network congestion [Collaborative-Cloud-Edge-Computing]. However, the ability of edge computing is limited by its processing power to perform resource intensive tasks. Thereby, a joint hierarchical architecture consisting of collaborative design involving user, edge, and the cloud is required to reduce the end-to-end latency and energy consumption and provide optimal computing performance.

When the user’s device cannot process any task on its own due to processing delay or computational limitations, it offloads the task to the edge. The edge task module will decide if the task would be processed locally at the edge or processed collaboratively with the upper cloud. Proportional resources are allocated for the network slices in the edge and the cloud. Here, the split of the task between the cloud and the edge would be decided by the task module in the edge. Several strategies for task offloading can be used, such as taking into account the service time for edge and the cloud and offloading the task to the upper layers in such a way to maximize parallelism among the user, edge, and cloud [Concurrent-tasks-offloading]. Besides, strategies based on energy consumption minimization and maximizing the throughput of the user [Offloading-services-for-mobile-devices] may be used for optimal resource allocation as well.

When a task requiring low computational processing is offloaded to the upper layers, the edge node can process it locally for the lower end-to-end delay and higher energy efficiency. However, when a computational-intensive task is relayed to the upper layers, instead of offloading the complete task to the cloud, the task module in the edge may split the task between the edge and the cloud. The split of tasks between the edge and cloud node may be based on the computational delay of the edge node, computational delay of the cloud node, and transmission delay of the cloud server.

The task module will track the resource utilization and the running applications and their performances. The resource management is done so that the QoS of the delay-sensitive traffic and resource utilization is maintained. It may have different sub-modules for task placement and scheduling by tracking the state of different tasks [iFogSim].

2.2. QoS based Resource management for Network Slicing

Using a Software-Defined Networking[SDN] based architecture can help manage the network slices centrally with optimized resource utilization and cost-efficiency. An efficient network slicing resource management is vital for latency-bound traffics. Dynamically allocating resources based on the traffic needs and priority would help manage the network in a more efficient and optimized manner. Therefore, implying that the VNs would be mapped based on the slice traffic and required QoS. For delay-sensitive traffic, the QoS can be based on the latency requirements, such as prioritizing delay-sensitive traffic as compared to delay-tolerant traffic such as live video vs. stored ones.

3. Example

One of the most time-sensitive XR applications includes healthcare, where a surgeon can utilize XR to perform surgery, even remotely. A collaborative edge and cloud model is highly desirable for both benefits of low-latency and high throughput. For such use-cases, the network needs to deliver data with low latency and high reliability. The application can sense and deliver the correct field of view while minimizing the motion-to-photon latency. In this regard, having network slicing priority can help prioritize the traffic for such cases, whereas having a collaborative edge-cloud model can reduce the latency since processing the task at the central cloud is not advised as this would increase the motion-to-photon latency. For real-time video processing such a remote surgery, applications require combining and synchronizing real-world data with the user’s motion, thereby requiring a massive rendering process. Since these graphics require heavy rending, the task is augmented by the upper edge and

cloud layers. When the user device is not able to process the incoming video frames due to its limited processing capabilities, it offloads the task to the edge, the edge, instead of offloading the whole frame to the cloud, offloads only difference of the frame compared to the previous frame [Collaborative-Edge-and-Cloud] for processing. Thereby sending only the frame difference instead of the whole frame, hence minimizing the latency and saving bandwidth.

4. IANA Considerations

This document has no actions for IANA.

5. Security Considerations

Security aspects relative to network slices (e.g., for transport slices, in [I-D.liu-teas-transport-network-slice-yang]) are applicable.

6. Acknowledgment

The author would like to thank Spencer Dawkins for reviewing the draft.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[Collaborative-Cloud-Edge-Computing] "Collaborative Cloud and Edge Computing for Latency Minimization", <https://rb.gy/sf2ctz>.

[Collaborative-Edge-and-Cloud] "Collaborative Edge and Cloud Neural Networks for Real-Time Video Processing", <http://www.vldb.org/pvldb/vol11/p2046-grulich.pdf>.

[Concurrent-tasks-offloading] "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing", <https://ieeexplore.ieee.org/document/6849257>.

[draft-ietf-mops-ar-use-case-00]
Krishna, R. and A. Rahman, "Media Operations Use Case for an Augmented Reality Application on Edge Computing Infrastructure", draft-ietf-mops-ar-use-case-00 (work in progress), March 2021.

[I-D.draft-defoy-coms-subnet-interconnection-04]
de Foy, X., Rahman, A., Galis, A., Makhijani, K., Qiang, Li., Homma, S., and P. Martinez-Julia, "Interconnecting (or Stitching) Network Slice Subnets", draft-defoy-coms-subnet-interconnection-04 (work in progress), March 2020.

[iFogSim] "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments", <https://arxiv.org/abs/1606.02007>.

[Latency-Network-AR-VR]
"Support Precise Latency for Network Based AR/VR Applications with New IP", <https://eprints.eudl.eu/id/eprint/856/1/eai.27-8-2020.2294291.pdf>.

[Offloading-services-for-mobile-devices]
"On effective offloading services for resource-constrained mobile devices running heavier mobile Internet application", <https://rb.gy/zdvwvv>.

[SDN] ""Software-defined networking." Communications of the ACM 56.9 (2013): 16-19", <https://dl.acm.org/doi/fullHtml/10.1145/2500468.2500473>.

[Spartacus]
"Spartacus: Service priority adaptiveness for emergency traffic in smart cities using software-defined networking", <https://rb.gy/cw0pbc>.

Author’s Address

Rohit Abhishek
Tencent
2747 Park Blvd
Palo Alto, California 94306
United States

Phone: 8165857500
Email: rabhishek@rabhishek.com