Experimental Evaluation of the Message Formats' Impact for Communication in Multi-party Edge Computing Applications

Tarciso Braz de Oliveira Filho[1][a], Dalton Cézane Gomes Valadares[1,2][b], Thiago Fonseca Meneses[1], Adauto Ferreira de Barros[1], Aramis Sales Araujo[1] and Danilo F. S. Santos[1]

[1]*Federal University of Campina Grande, VIRTUS - Research, Development and Innovation Center, Campina Grande, PB, Brazil*

[2]*Federal Institute of Pernambuco, Caruaru, PE, Brazil*

Keywords: Message Formats, Edge Computing, Video Inference Applications.

Abstract: The increasing integration of 5G, multi-access edge computing (MEC), and microservices, benefits the development of applications that demand low coupling, low communication latency, high scalability, and high availability. An usual scenario that deals with such requirements is a video application, either to process inference on video images or process video analytics. Given that video data are considered heavy to process and transmit, we should investigate the best way to handle such data. This work presents an experimental setup for the comparison between four data formats used to send video frames among distributed application components in a MEC server. We measured and analyzed the communication latency when sending video data between distributed parties, considering three scenarios.

1 INTRODUCTION

Currently, we have a trend of deploying the distributed systems' architectures among different microservices. Microservices are services with specific purposes, developed and deployed independently, allowing an easier provision and management of systems that require high concurrency, high availability, high scalability, and low coupling (Liu et al., 2020). Many application providers deploy their microservices in cloud servers, but the deployment possibilities are growing with the increasing adoption of fog and edge computing (Zou et al., 2019; Sunyaev, 2020). Besides, related to the edge computing concept, there is still the multi-access edge computing (MEC, previously called mobile edge computing) (Filali et al., 2020), which enables the cloud/edge benefits for telecommunication operators. Thus, with the adoption of the 5G and its enabling technologies (e.g., network function virtualization and network slices), the application providers begin to deploy their microservices also at MEC servers.

The power of 5G, MEC, and microservices integration becomes an enabler to the many applications that demand video processing with strict requirements, such as low communication latency and high video quality. For instance, a process that can benefit from this integration is automatic video tagging, which assigns meaningful human-friendly information to video frames (Arca et al., 2020). Another application that benefits from those technologies is video inferencing, such as a monitoring system that performs pedestrian recognition (Xu et al., 2020).

Given the vast possibility of different applications, developers can use many possible technologies and tools for developing and integrating such applications. Although the distributed systems can interoperate in diverse ways, a concern arises when efficient communication is required: what is the most suitable data format to send video data (frames) among distributed components?

We decided to answer that question as a project decision once we are developing a platform to help in the deployment process of artificial intelligence applications in MEC servers. As a proof of concept, we have implemented a video inference application to work as one of the possible use cases. During the development, we faced some communication problems when transmitting and processing the video frames. For this reason, we created a testbed to evaluate the communication between distributed compo-

[a] https://orcid.org/0000-0001-8620-3877
[b] https://orcid.org/0000-0003-1709-0404

Filho, T., Valadares, D., Meneses, T., Ferreira de Barros, A., Araujo, A. and Santos, D.
Experimental Evaluation of the Message Formats' Impact for Communication in Multi-party Edge Computing Applications.
DOI: 10.5220/0010717700003058
In *Proceedings of the 17th International Conference on Web Information Systems and Technologies (WEBIST 2021)*, pages 534-543
ISBN: 978-989-758-536-4; ISSN: 2184-3252

nents of the video inference application, deploying them in the MEC server. We performed experiments to measure the communication latency when considering different scenarios and data formats to send the video frames.

To guide the investigation to answer the research questions, we defined the business and technical problems as follow:

- Business problem: Investigate means of transmitting and receiving video data efficiently among distributed components in a MEC server;
- Technical problems:
  - Implement a testbed considering the communication of video data among distributed components using different data formats;
  - Measure and compare the communication latency among the distributed components considering distinct scenarios and data formats.

The main contributions of this paper are:

- a framework to evaluate communication between distributed components, considering different message formats. This framework can help when other formats and scenarios need evaluation, or other metrics are necessary, allowing the developers to benchmark the interested metrics;
- a framework's implementation, considering a distributed video inference application and different deployments in a MEC server;
- a comparison of four message formats for exchanging JPEG-compressed images between the components of a distributed video inference application. For this comparison, the experiments considered the following formats: byte array, base64-encoded byte array, base64-encoded string, and JSON-encapsulated base64-encoded string;

The remainder of this paper follows this structure: Section 2 presents the background, which briefly describes the gRPC technology and usual message formats generally used to exchange data among different applications; Section 3 presents some works related to the content of this paper; Section 4 describes the methodological design defined to guide the execution of the experiments; Section 5 exhibits and discuss the obtained results; and Section 6 finally concludes this paper, presenting a summary and suggesting future work.

2 BACKGROUND

2.1 Multi-access Edge Computing

Multi-access edge computing (MEC) refers to applications that enable the provision of telecommunications and IT services, resources such as storage and computing, whose benefits range from low latency to optimized use in the mobile backhaul of network operators, providing cloud computing benefits at the edge of the RAN (Radio Access Network) (Taleb et al., 2017).

The use of 5G technology will bring an unimaginable increase in traffic volume of networks and computing demands, due to the emergence of new compute-intensive applications and the Internet of Things (IoT). According to Pam et al. (Pham et al., 2020) how to perform intensive computing on end-users, those with limited devices, has recently become a natural concern. Thus, MEC emerges as a key technology in the emerging fifth-generation (5G) network, hosting computationally intensive applications, processing large volumes of data close to users, which will enable applications such as driverless vehicles, VR/AR, robotics, and immersive media. Despite this, the use of the MEC is still in its infancy and requires a constant effort from the academic and industrial communities (Pham et al., 2020).

2.2 Video Analytics

Video analysis is a subset of computer vision and of artificial intelligence that analyze images from a camera to recognize humans, vehicles, objects, and events. It has advancement, but they have come at the computing and network cost. Low-latency video analytic is becoming increasingly important in security and smart city scenarios (Yi et al., 2017).

According to Pasandi (Pasandi and Nadeem, 2020) more and more organizations are using video analytics, deploying hundreds of cameras for various types of applications, from monitoring industrial or agricultural sites to retail.

As Zhang et al. (Zhang et al., 2017), it is necessary to develop efficient components to use the data power offered by the cameras. The use of edge computing, where thousands of devices will be connected and streaming video, brings the challenge of reducing bandwidth and response time. The solution for large-scale real-time video analytic currently uses distributed architectures using public cloud, but with trade-offs in quality, delay and reconfiguration (Ananthanarayanan et al., 2017).

2.3 GRPC

GRPC[1] is a open-source Remote Procedure Call (RPC) framework that aims to connect applications with a fast and sleek communication independent of running environment. Bringing scalability and a simple service definition with bi-directional streaming for a server-client structure.

For communication purposes, gRPC allows different parts of the system to exchange data that is essential for the application to function. With gRPC, a component is able to delegate the processing of part of its flow to a specialized module that integrates the system, creating a distributed network of processes. In this sense, it is necessary that the communication is well defined and structured, allowing both parties to be aware of what will be necessary for implementation and what is the expected result of this process. To achieve this, an Interface Definition Language (IDL) is used, which will define what can be accessed by the client module, what must be consumed by the server module and, finally, what will be delivered at the end of the execution of the process.

Normally, gRPC makes use of Protocol Buffer, an IDL that performs the serialization and structuring of transported data. This ensures that the service has a well-defined interface and enables communication in a language-agnostic and platform-neutral way (Indrasiri and Kuruppu, 2020). With this definition, the server and client code are generated, making it easier for the developer to identify how the communication between the different platforms will be carried out (Araújo et al., 2020). Both generated codes work similarly to a local call and gRPC handles the complexity needed to carry out this communication.

In the context of Edge Computing, gRPC becomes an interesting solution, providing an efficient architecture with low resource use of the embedded systems involved, as evaluated by Araújo et al. (Araújo et al., 2020). Therefore, with the need for communication between different application modules, the choice for gRPC has become a trend in the development of distributed technologies (Indrasiri and Kuruppu, 2020).

3 RELATED WORK

Although we have not found works comparing data formats, we can see works highlighting specific formats, whether exploring their capabilities or proposing improvements. For instance, Taktek et al. (Taktek et al., 2018) compare two schemes for labeling operation in native XML databases. For the comparison, the authors employed UTF-8, UTF-16, and UTF-23 for the encoding and decoding processes. Psaila et al. (Psaila. et al., 2020) presented a model to handle GeoJSON documents.

Proos et al. compared three messaging protocols considering a scenario of Internet of Vehicles and two serialization formats (Protobuf and Flatbuffers). Regarding the serialization formats, they concluded that Protobuf, which is used by gRPC, has a faster serialization speed and smaller serialized message size.

With the use of MEC, the academic community and industry seek to solve problems with workloads and application response time at the edge. Randazzo and Tinnirello (Randazzo and Tinnirello, 2019) propose that a standardized architecture must be designed to perform the migration of services to the MEC in a safe and timely manner. He assesses that the final solution can be architecture with the advantages of using VM, in terms of security, and containers related to performance in the data processing.

According to Sawant (Sawant, 2019), the exponential increase in data volume is characterized by heterogeneous formats and disparate data sources. As a result, the task of transmitting uniform data to analysis systems becomes complex, increasing the complexity of current systems.

4 EXPERIMENTAL DESIGN

The experiments were run in different scenarios, varying the pipeline architecture and the image data format exchanged between client and server. Several time measurements were performed to describe the latency of each step of the pipeline, being able to perform a deeper analysis. The scenarios are described in detail below.

4.1 Architectural Designs

There are three basic components comprising the experiment architecture:

- the RTSP Video Stream, which combines a ffmpeg[2] video stream and a Real-Time Streaming Protocol - RTSP(Rao et al., 1998) server
- the gRPC-MLInference Client, which queries the server with images to perform inference on, composed of:

[1] https://grpc.io

[2] http://ffmpeg.org/

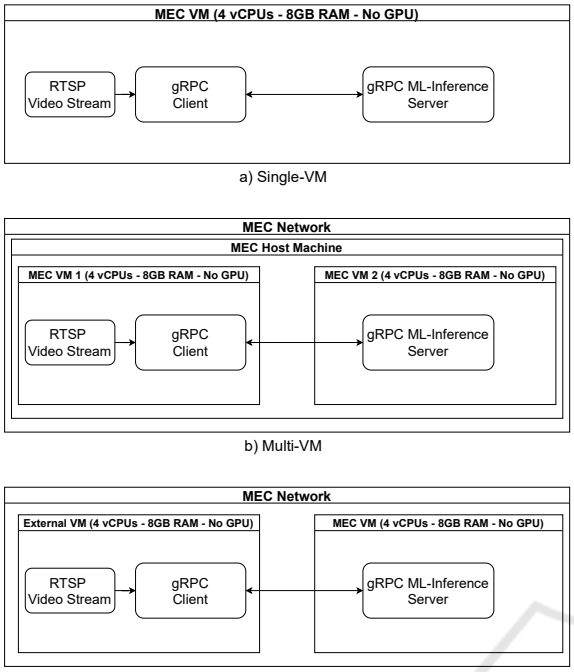

Figure 1: Experiment Architectural Designs: a) Single-VM, b) Multi-VM, c) Remote MEC.

  - VideoHUB, responsible for retrieving video frames from the RTSP stream and preparing them for inference
  - DemoApp, responsible for receiving and displaying the inference results
- the gRPC-MLInference Server, which receives frames and performs Object Detection inference on them using an OpenVINO-optimized[3] SSD MobileNET ML model (Sandler et al., 2018).

The scenarios were executed in a Nokia Airframe server powered by the Intel® Xeon® processor E5-2600 v3 family with up to 14 cores per socket. We used Openstack as an IaaS platform, provisioning at all four virtual machines each with 4 vCPU and 8GB of RAM in a MEC server and one external virtual machine with 4 vCPU and 8GB of RAM.

4.1.1 MEC Single-VM

The Single-VM architectural design consists of the deployment of both gRPC-MLInference client and server on the same machine. In this experiment, we used a MEC infrastructure, deploying the components on a single MEC virtual machine, as depicted in Figure 1-a.

[3]https://docs.openvinotoolkit.org/

Figure 2: Experiment Input Video Screenshot.

4.1.2 MEC Multi-VM

The Multi-VM architectural design consists of the deployment of gRPC-MLInference client and server on different machines within MEC. In this case, components were deployed on separate MEC virtual machines, as can be seen in Figure 1-b.

4.1.3 Remote MEC

The Remote MEC architectural design consists of the deployment of the gRPC-MLInference server on a MEC machine, and the client on an external machine within the same network (see Figure 1-c).

4.2 Data Formats

The second factor of variation in the experiment is the format of the data exchanged between the client and server through gRPC. The hypothesis is that the data format and its inherent attributes affect the transmission speed and, therefore, the latency. The video used as input for the experiment had a FULL-HD Resolution (1080p) encoded in H.264 portraying a car trip in the city of São Paulo, Brazil, as depicted in Figure 2.

To access the baseline transmission time measurements for the chosen protocol in this experiment, a base message consisted of a single Boolean value. As for the transmission of video frames, a JPEG encoding was chosen, paired with Base64 encoding combinations as shown below.

- JPEG-compressed image
  - JPEG image byte array
  - Base64-encoded JPEG image byte array
  - Base64-encoded JPEG image string
  - JSON-encapsulated Base64-encoded JPEG image string

These particular combinations have been chosen for their popularity in web applications and to introduce a variation in data size and encoding/decoding complexity.

4.3 Latency Measurements

As a means to perform an effective evaluation of the performance of the Inference Pipeline with respect to time, we define a set of latency measurements, collected for each combination of the experiment variables. Each measurement is described in detail below. Their time unit is milliseconds (ms).

- Client Frame Handling - time taken for pre-processing of the original frame and post-processing of the inference frame on the client
- Server Frame Handling - time taken for pre-processing of the original frame and post-processing of the inference frame on the server
- Round Transmission Time (RTT) - time taken for the original frame to go from the client to the server and for the inferred frame to go from the server to the client
- Inference Time - time taken for the server to perform inference on the original frame

4.4 Aggregation Functions Applied

In order to have a consistent evaluation, the frame inference pipeline is run 1000 rounds for each experimental setup variables (architecture and data format) combination, collecting the values for each latency measurement. Finally, the values were averaged using the aggregation functions described below.

- Median - characterizes the average value of each latency measurement
- Standard Deviation - characterizes the variance observed in the values of each latency measurement

5 RESULTS AND DISCUSSION

5.1 Average Latency Measurements Observed

Initially, we analyze the average (median) latency measurements for each experimental design scenario, as described in the previous section.

5.1.1 MEC Single-VM

The median values of each latency measurement obtained from the experiment round on the MEC Single-VM architecture can be seen in Table 1. By analyzing them, we notice that:

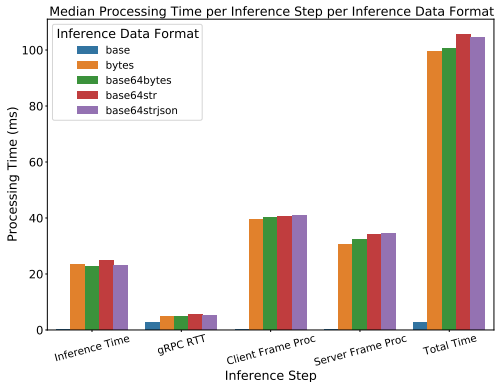

Figure 3: MEC Single-VM Median Latency Measurements per Inference Data Format.

- gRPC Round Transmission Time is around 2.5 ms for the base message and increases to 5.5 ms in the slowest setting (Base64-encoded string frame format). As the architectural design in Single-VM, this is the time the message takes to hit the network card and fall back to the server process port.
- Client frame processing time is around 40 ms and Server frame processing time is between 30 ms and 34 ms. These steps dominate the latency, signaling there might be some improvement by investing in data formats which simplify frame encoding and decoding.
- Inference Time ranges between 22 ms and 25 ms (in an 8GB-RAM/4-vCPU virtual machine), keeping in mind that both client and server are using the same computing and memory resources.
- Total Time stays between 99 ms and 105 ms, providing an output video of around 10 fps.

A visual comparison of the latency measurements observed on the MEC Single-VM architecture for the different data formats can be seen in Figure 3. By examining it, we conclude that:

- Although the difference observed in latency among the data formats is not large, the bytes data format presents the overall lowest latency measurements among the experimented data formats, contributing to the hypothesis that raw formats promote better performance due to the lack of pre/post-processing.
- Overall, the different data formats behave very similarly in terms of latency for each inference pipeline step.

5.1.2 MEC Multi-VM

The median values of each latency measurement obtained from the experiment round on the MEC Multi-

Table 1: MEC Single-VM Median Latency Measurements.

Data Format	Client Frame Proc	gRPC RTT	Inference Time	Server Frame Proc	Total Time
base	0.02	2.55	0.0	0.01	2.58
base64bytes	40.21	4.64	22.68	32.18	100.67
base64str	40.45	5.5	24.76	34.11	105.73
base64strjson	40.89	5.29	22.87	34.31	104.32
bytes	39.59	4.66	23.54	30.6	99.42

Table 2: MEC Multi-VM Median Latency Measurements.

Data Format	Client Frame Proc	gRPC RTT	Inference Time	Server Frame Proc	Total Time
base	0.18	3.18	0.0	0.0	3.19
base64bytes	4.66	769.24	2.76	3.39	769.48
base64str	4.31	769.14	3.25	3.05	769.54
base64strjson	4.7	940.51	3.7	3.5	940.41
bytes	4.24	545.28	4.16	3.46	544.9

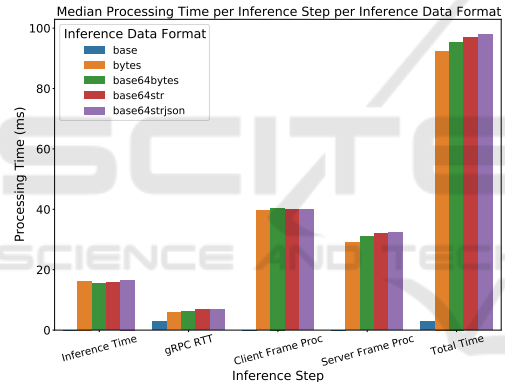

Figure 4: MEC Multi-VM Median Latency Measurements per Inference Data Format.

VM architecture can be seen in Table 2. By analyzing them, we notice that:

- gRPC Round Transmission Time does not increase much when moving from a single-vm to a multi-vm architecture, reaching at most 6.8 ms on the slowest data format (Base64 String JSON), which indicates the communication between VMs within the same host is very efficient, not causing significant impact on the inference pipeline performance.
- Client and Server frame processing time presents a very similar range as the one observed in the Single-VM architecture.
- Inference Time is lower when compared to the Single-VM architecture, taking between 15 ms and 16 ms (reduction of 7-9 ms). This might be due to the separation of client and server in different environments, not sharing resources, leaving more resources for the inference (in an 8GB-RAM/4-vCPU virtual machine).
- Total Time decreases as well, pulled by the lower inference time, providing an output video with a bit more than 10 fps.

A visual comparison of the latency measurements observed on the MEC Multi-VM architecture for the different data formats can be seen in Figure 4. By examining it, we conclude that:

- Following the trend of the Single-VM Architecture, the difference observed in latency among the data formats is not large, but the bytes data format presents the overall lowest latency measurements among the experimented data formats.
- Again, the different data formats behave very similarly in terms of latency for each inference pipeline step.

5.1.3 Remote MEC

The median values of each latency measurement obtained from the experiment round on the Remote MEC architecture can be seen in Table 3. By analyzing them, we notice that:

- gRPC Round Transmission Time increases drastically (around 5 times) from the values observed in the previous rounds, reaching up to 23 ms on the slowest data format (Base64 String JSON), which indicates the communication between VMs in different hosts and, thus, subject to network traffic

Table 3: Remote MEC Median Latency Measurements.

Data Format	Client Frame Proc	gRPC RTT	Inference Time	Server Frame Proc	Total Time
base	0.02	3.83	0.0	0.0	3.86
base64bytes	31.53	21.4	16.63	29.68	102.42
base64str	31.69	22.84	16.75	31.01	105.54
base64strjson	32.35	23.39	16.81	31.58	107.79
bytes	30.75	19.67	17.23	28.23	101.39

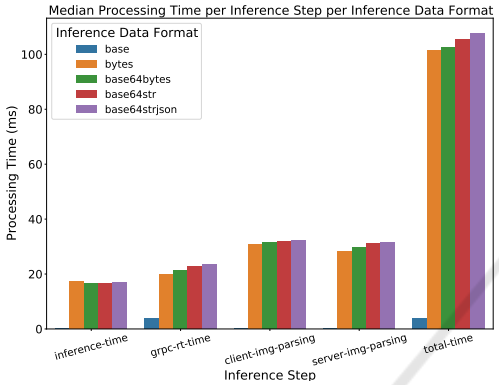

Figure 5: Remote MEC Median Latency Measurements per Inference Data Format.

and anomalies decreases inference pipeline performance.

- Server frame processing time presents a very similar range as the one observed in the Single and Multi-VM architecture. Client frame processing, in turn, is lower, probably due to the fact that client VM host has robust computing specs.
- Inference Time is very similar to the observed in Multi-VM architecture, contributing to the hypothesis that due to the separation of client and server in different environments, not sharing resources, there are more resources for the inference in the server VM.
- Total Time increases, pulled by the higher gRPC Round Transmission Time, providing an output video with a bit less than 10 fps.

A visual comparison of the latency measurements observed on the MEC Multi-VM architecture for the different data formats can be seen in Figure 5. By examining it, we conclude that:

- Following the trend of both previous architectures, the difference observed in latency among the data formats is not large, but the bytes data format presents the overall lowest latency measurements among the experimented data formats.
- Once more, the different data formats behave very similarly in terms of latency for each inference pipeline step.

5.1.4 Discussion

Analyzing the results obtained for the different scenarios, we can conclude:

- gRPC behaves well in a controlled network environment, but is dramatically affected by network traffic.
- gRPC communication performance is not significantly decreased if pipeline components are in separated VMs in the same host (MEC host in this case).
- Inference time benefits from a robust or lightly-loaded infrastructure.
- In all evaluated scenarios, the client and server frame processing time dominated the latency, indicating there might be some room from improvement on these steps of the pipeline.
- In all evaluated scenarios, the raw bytes data format presents slightly better performance than the others.

5.2 Variability Observed in Latency Measurements

Finally, we analyze the latency measurements for each experimental design scenario, as described in the previous section.

5.2.1 MEC Single-VM

Each latency measurement obtained from the experiment round on the MEC Single-VM architecture can be seen in Table 4. By analyzing them, we notice that all latency measurements in all formats are reasonably low, which indicates this scenario provides high stability to the pipeline performance.

A visual comparison of the latency measurements on the MEC Single-VM architecture for the different

Table 4: MEC Single-VM Latency Measurements (in ms).

Data Format	Client Frame Proc	gRPC RTT	Inference Time	Server Frame Proc	Total Time
base	0.01	1.91	0.0	0.0	1.91
base64bytes	4.94	2.2	6.37	4.5	11.5
base64str	4.3	3.26	6.24	3.74	10.36
base64strjson	4.33	2.83	6.54	4.24	11.72
bytes	4.25	2.65	6.01	3.79	10.1

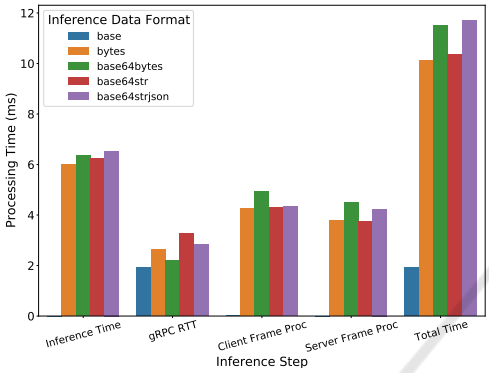

Figure 6: MEC Single-VM Latency Measurements per Inference Data Format.

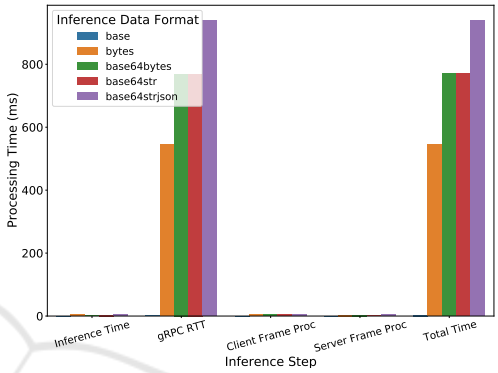

Figure 7: MEC Multi-VM Latency Measurements per Inference Data Format.

data formats can be seen in Figure 6. By examining it, we conclude that the difference observed in latency among the data formats is not large, but the bytes data format presents the overall lowest latency measurements among the experimented data formats. In addition, the different data formats behave very similarly in terms of latency variability for each inference pipeline step.

5.2.2 MEC Multi-VM

The latency measurements obtained from the experiment round on the MEC Multi-VM architecture can be seen in Table 5. By analyzing them, we notice that all latency measurements are reasonably low, except for the gRPC Round Transmission Time, which ranges from 545 ms (raw bytes format) to 940 ms (Base64 String JSON format), indicating some variability (instability) in the pipeline performance for this scenario which cannot be seen in the chosen average latency metric.

A visual comparison of the latency measurements on the MEC Multi-VM architecture for the different data formats can be seen in Figure 7. By examining it, we conclude that:

- The gRPC Round Transmission Time dominates the total latency of the pipeline measurements.

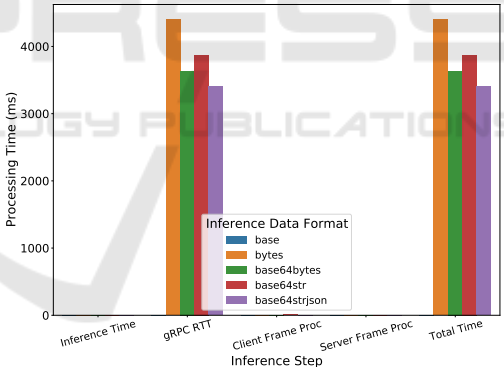

Figure 8: Remote MEC Latency Measurements per Inference Data Format.

- The bytes data format presents the overall lowest latency measurements among the experimented data formats.
- The different data formats behave very similarly in terms of latency variability for each inference pipeline step.

5.2.3 Remote MEC

Each latency measurement obtained from the experiment round on the Remote MEC architecture can be seen in Table 6. By analyzing them, we notice that once more the latency of the gRPC Round Transmis-

Table 5: MEC Multi-VM Latency Measurements (in ms).

Data Format	Client Frame Proc	gRPC RTT	Inference Time	Server Frame Proc	Total Time
base	0.18	3.18	0.0	0.0	3.19
base64bytes	4.66	769.24	2.76	3.39	769.48
base64str	4.31	769.14	3.25	3.05	769.54
base64strjson	4.7	940.51	3.7	3.5	940.41
bytes	4.24	545.28	4.16	3.46	544.9

Table 6: Remote MEC Latency Measurements (in ms).

Data Format	Client Frame Proc	gRPC RTT	Inference Time	Server Frame Proc	Total Time
base	0.02	1.26	0.0	0.0	1.26
base64bytes	6.31	3634.34	2.81	3.72	3634.58
base64str	12.0	3875.19	3.22	3.45	3876.23
base64strjson	7.82	3403.95	3.89	3.33	3404.0
bytes	8.2	4403.59	3.33	3.32	4404.22

sion Time pops up, this time even higher than in the Multi-VM architecture, now ranging from 3400 ms (Base64 String JSON format) to 4403 ms (raw bytes format), indicating high variability (instability) in the pipeline performance for this scenario which cannot be seen in the chosen average latency metric.

A visual comparison of the latency measurements on the Remote MEC architecture for the different data formats can be seen in Figure 8. By examining it, we conclude that:

- The gRPC Round Transmission Time dominates the total latency of the pipeline measurements.
- Surprisingly, the Base64 String JSON format presents the overall lowest latency measurements among the experimented data formats, with the raw bytes format presenting the highest latency values.
- The different data formats behave very similarly in terms of latency variability for each inference pipeline step.

5.2.4 Discussion

Analyzing the results obtained for the latency metric in the different scenarios, we can conclude:

- gRPC is very stable when not using network, but can become dramatically unstable when subject to network traffic (even in the same host).
- Except for gRPC Round Transmission Time, all latency measurements present reasonably low variability (high stability) for all evaluated scenarios.

6 CONCLUSION

Motivated by a project decision when developing a video application, which should be deployed in a MEC server, we decided to analyze different data formats to transmit video frames. Thus, we implemented a testbed in the MEC server, considering three different scenarios for the distributed components of the application. For each scenario, we performed experiments considering the communication among the components when transmitting video frames. For each communication cycle, we measured the latency. The experiments considered four data formats for the video frames: bytes, base64 bytes, base64 string, and base64 string JSON. We also measured the communication latency during the gRPC use. The results demonstrate that all the formats present similar latency measurements with a slight vantage for the bytes format in two of the three scenarios. We can also conclude that gRPC is very stable for scenarios without network traffic.

As future work, we suggest:

- Run the experiments varying the scenarios (e.g., using another deployment variations for the components and other data formats);
- Investigate ways to make the pre and post-processing of video frames (encoding/decoding) more efficient once this is the operation dominating the communication latency:
- Understand what makes gRPC unstable when subject to the network traffic (even inside the same host);

- Perform experiments running the components inside VMs with more CPU and RAM to analyze the impact in the frame processing and inference process.

ACKNOWLEDGEMENTS

We thank the Center of Research, Development, and Innovation for Information, Communication, and Automation Technologies - VIRTUS/UFCG for supporting this work's development. This work was partially financed by the 3rd RDI Agreement between SOFTEX and UFCG - Project Edge Framework.

REFERENCES

Ananthanarayanan, G., Bahl, P., Bodík, P., Chintalapudi, K., Philipose, M., Ravindranath, L., and Sinha, S. (2017). Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67.

Araújo, M., Maia, M. E. F., Rego, P. A. L., and De Souza, J. N. (2020). Performance analysis of computational offloading on embedded platforms using the gRPC framework. In Mena, F. M., Yucatan, U. A. D., Mexico, Duarte, E., of Parana, F. U., and Brazil, editors, *8th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2020)*, pages 1–8, Cancún, Mexico. Candy E. Sansores, Universidad del Caribe, Mexico, Nazim Agoulmine, IBISC Lab, University of Evry - Paris-Saclay University.

Arca, A., Carta, S., Giuliani, A., Stanciu, M., and Recupero, D. (2020). Automated tag enrichment by semantically related trends. In Marchiori, M., Mayo, F., and Filipe, J., editors, *WEBIST 2020 - Proceedings of the 16th International Conference on Web Information Systems and Technologies*, pages 183–193. SciTePress.

Filali, A., Abouaomar, A., Cherkaoui, S., Kobbane, A., and Guizani, M. (2020). Multi-access edge computing: A survey. *IEEE Access*, 8:197017–197046.

Indrasiri, K. and Kuruppu, D. (2020). *gRPC: up and running: building cloud native applications with Go and Java for Docker and Kubernetes*. O'Reilly Media.

Liu, G., Huang, B., Liang, Z., Qin, M., Zhou, H., and Li, Z. (2020). Microservices: architecture, container, and challenges. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 629–635.

Pasandi, H. B. and Nadeem, T. (2020). Convince: Collaborative cross-camera video analytics at the edge. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–5.

Pham, Q.-V., Fang, F., Ha, V. N., Piran, M. J., Le, M., Le, L. B., Hwang, W.-J., and Ding, Z. (2020). A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art. *IEEE Access*, 8:116974–117017.

Psaila., G., Marrara., S., and Fosci., P. (2020). Soft querying geojson documents within the j-co framework. In *Proceedings of the 16th International Conference on Web Information Systems and Technologies - WEBIST,*, pages 253–265. INSTICC, SciTePress.

Randazzo, A. and Tinnirello, I. (2019). Kata containers: An emerging architecture for enabling mec services in fast and secure way. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 209–214.

Rao, A., Lanphier, R., and Schulzrinne, H. (1998). Real Time Streaming Protocol (RTSP). RFC 2326.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. pages 4510–4520.

Sawant, O. V. (2019). Combating dirty data using data virtualization. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pages 1–5.

Sunyaev, A. (2020). *Fog and Edge Computing*, pages 237–264. Springer International Publishing, Cham.

Taktek., E., Thakker., D., and Neagu., D. (2018). Comparison between range-based and prefix dewey encoding. In *Proceedings of the 14th International Conference on Web Information Systems and Technologies - WEBIST,*, pages 364–368. INSTICC, SciTePress.

Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., and Sabella, D. (2017). On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys Tutorials*, 19(3):1657–1681.

Xu, Z., Wu, J., Xia, Q., Zhou, P., Ren, J., and Liang, H. (2020). Identity-aware attribute recognition via real-time distributed inference in mobile edge clouds. MM '20, page 3265–3273, New York, NY, USA. Association for Computing Machinery.

Yi, S., Hao, Z., Zhang, Q., Zhang, Q., Shi, W., and Li, Q. (2017). Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, SEC '17, New York, NY, USA. Association for Computing Machinery.

Zhang, H., Ananthanarayanan, G., Bodik, P., Philipose, M., Bahl, P., and Freedman, M. J. (2017). Live video analytics at scale with approximation and delay-tolerance. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, page 377–392, USA. USENIX Association.

Zou, Z., Jin, Y., Nevalainen, P., Huan, Y., Heikkonen, J., and Westerlund, T. (2019). Edge and fog computing enabled ai for iot-an overview. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 51–56.