# Master Thesis Topic: Evaluation of Computation Offloading Decision Making Model in BBR-enabled Edge Computing

**Name:** Rohit Jain

**Matriculation Number:** 2512376

**Department:** FB20

**Start date:** TBD

**Supervisors:** [1] Prof. Dr. Max Mühlhäuser, [2] Florian Brandherm

## Introduction & Motivation:

TCP congestion control has been historically a problem of great interest for the Internet since 1980s [6]. Today's most widely used congestion control mechanism – TCP CUBIC relies on loss of packets to identify congestion problem. However, this technique does not scale well with network throughput for varying large and small buffer queue sizes. In 2016 Google came up with idea of Bottleneck Bandwidth and Round-trip propagation time (BBR), which used a different approach – monitoring maximum recent network bandwidth and minimum recent round-trip delay to make a model, which helps to decide the optimal sending data rate and amount of data in flight at every moment during transmission, thereby reducing overall network congestion to a minimum [4].

This new protocol projected promising results, and therefore it gathered much interest from a wide range of researchers and developers' community for further testing (benchmarking and comparisons) and application (Wired and Wireless Networks) and also led to advancement in different forms of variations like BBR+, BBR v2, BBRfi.

The characteristics of BBR makes it relevant for application in Edge Computing as one of core goals for achieving low latency communication between Edge Client, Edge Server and/or Remote Server / Cloud Service. At the moment most edge computing devices use TCP CUBIC as their congestion control mechanism for both wired and wireless communication [7], so there is a great interest in the research community to experiment with different methods techniques to optimize uplink and downlink communication in context of edge computing [8].

This along with decision model for computation offloading in Mobile Edge Computing represents a good research gap for further study.

## Goals and Approach:

In this Master's thesis I aim to test and evaluate the computational offloading decision making in context of image labelling, between edge client and remote server based on evaluation of Network Bandwidth and Round Trip Time (RTT), along with respective processing time at client and server. In order to build a decision model for computation offloading, I would define an SLA based time deadline for a set of imaging labeling data for both edge client and server. I would then use a series of experiments to analyze tradeoffs between a choice of various pre trained neural networks for image labelling in order to meet the deadline of SLA.

While most other research work with BBR focuses on TCP based, kernel level implementation through Google's open-source project [9], in my approach I want to make my own limited features set custom implementation of BBR for QUIC based messaging in User mode instead of Kernel mode. At the moment TCP BBR is not available for direct use as a feature within Android Kernel [10]. It has also been stated that TCP BBR doesn't provide much noticeable improvement over TCP CUBIC without handset specific – customized packet pacing control, because of hardware specific performance variations.

Also, with TCP a packet gets sent only when its sending buffer is full otherwise it waits. This problem can be avoided with UDP with sending of individual datagrams. By using QUIC we are able to provide congestion control feature for UDP.

Since a lot of research has already been done with performance evaluation of BBR with TCP – mostly using the standard implementation by Google, the BBR with QUIC presents good opportunity for exploration, since there are multiple implementations available for QUIC [11], so some of these can serve as source of ideas for personal use case implementation.

Eventually through multiple experimentations and their evaluations, I would summarize different strategies for executing a computation task within maximum allowed time boundary in context of edge computing, with choice of decision to offload between client and server, choice of algorithms / models on the respective side and provide evaluation of tradeoffs from different choices.

## Tools and Techniques:

I would be using Android phone – Google Pixel 6 as my wireless edge client and I would develop an Android APP using Android Studio for development on PC, and then test and deploy on phone at a later stage. For Edge Server, I would use Ubuntu / Debian based VM located at university premises or a Raspberry Pi / Ubuntu Laptop at home.

The Android app will provide a user interface programmed in Java and Kotlin to capture camera image of objects from Google Pixel phone and then perform two tasks in parallel: [1] upload it to server for image labelling and wait for its response [2] execute image labelling on phone itself. It will then record timestamp for both tasks.

The image labelling would be done through the use of pretrained neural models available in form of ready to use API with Android SDK and Python. Choices for different models is still under review and pending further literature research.

For implementing BBR based congestion control on edge client, I would be developing an Android native API library in C++ which would be transmitting data and messages over QUIC / UDP. A similar implementation would also be done on Ubuntu or Debian based Server side also in C++. Library would be providing background mechanisms for collecting and optimizing metrics of BBR and generate predictions for RTT.

Important noticeable difference between Client vs Server network traffic is that while Client sends larger size data of scaled camera images, Server just sends short message of text labels of result of classification plus Server processing time. Overall, this represents short bursts of intermittent data traffic and not continuous stream of data. More details about this are provided in "Evaluation" section.

For evaluation over wireless connectivity, I would be using phone over 4G LTE and WI-FI. Furthermore, there is going to variation in experimentation with less crowded area such as home and more crowded area like TU Stadtmitte Mensa.

For simulating a low latency scenario, I can connect phone and server (personal laptop or a Raspberry Pi) to home WI-FI without any extra devices connected.

In order to get more realistic measurements of RTT values, there is a possible option to have a remote cloud server deployed with AWS at a geographically far off location.

Alternatively, there is another option to make WIFI router with Raspberry Pi to be able to simulate packet delays and jitter in network traffic using Linux's built in tools by preparing Shell Script or running commands live over SSH during experimentation.

## Evaluation:

From the setup and implementation explained above there is going to be different phases and interdependent aspects of my experimentations and its evaluations. These are explained as follows:

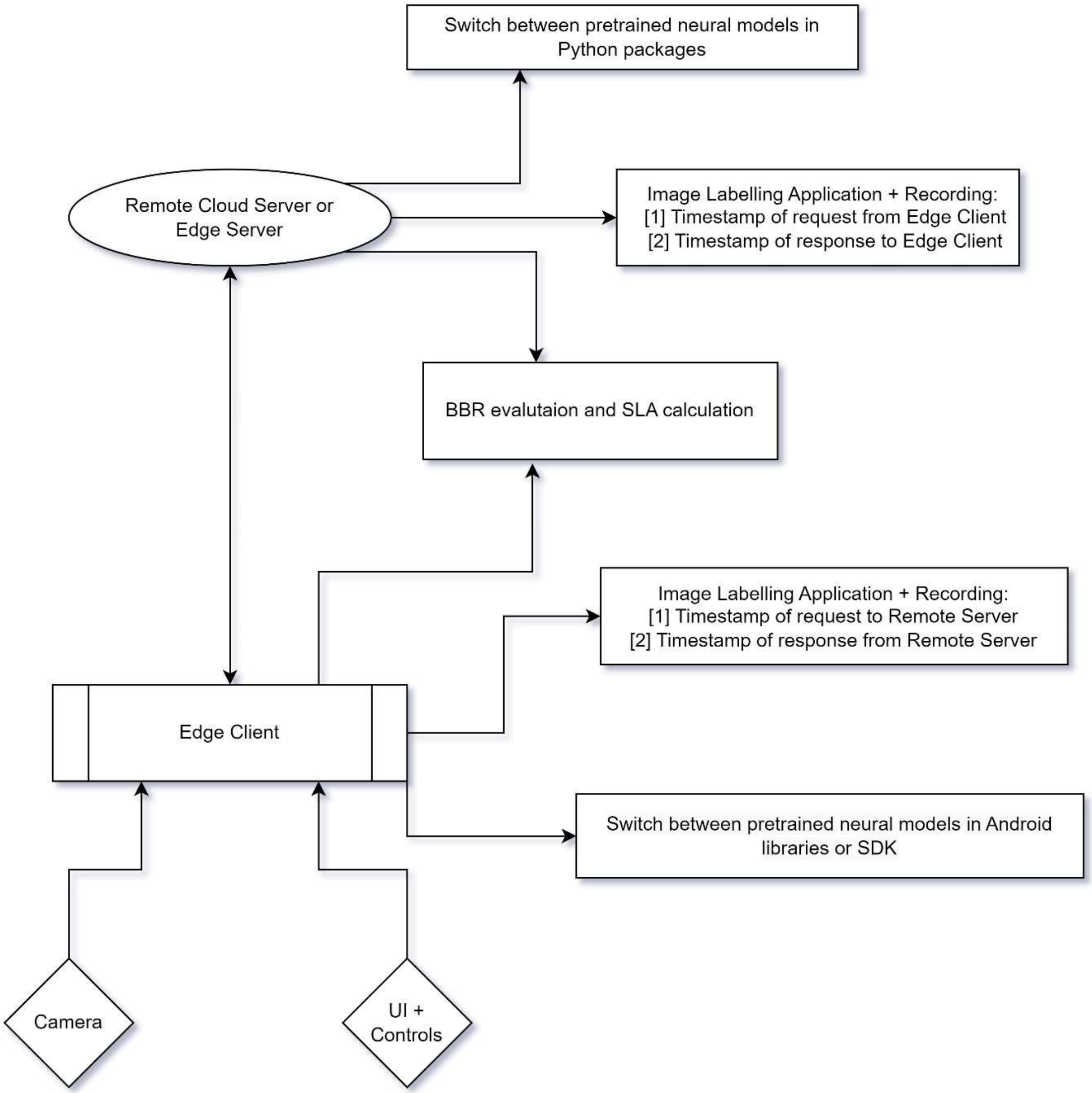
[1] **BBR performance evaluation with Wireless Networks**: I would be adjusting different parameters and experimental configurations of BBR [2] to observe Sending Rate & Throughput, Round Trip Time, number of Retransmissions and calculate Data in Flight. These parameters would be used to predict latency for upcoming transmissions and derive SLA parameters. While previously reviewed papers discuss about BBR behavior for continuous and long streams of data, it would be interesting to observe BBR behavior in our case, specifically for bursty and intermittent data traffic with larger size image from edge client and short texts message response from server.

[2] **Image labelling processing time**: Both on Edge client & Remote Server I would be recording processing time to calculate optimum time boundary for SLA of image labelling task. Time boundary may need to be adaptable depending on current and predicted values of latency in network.

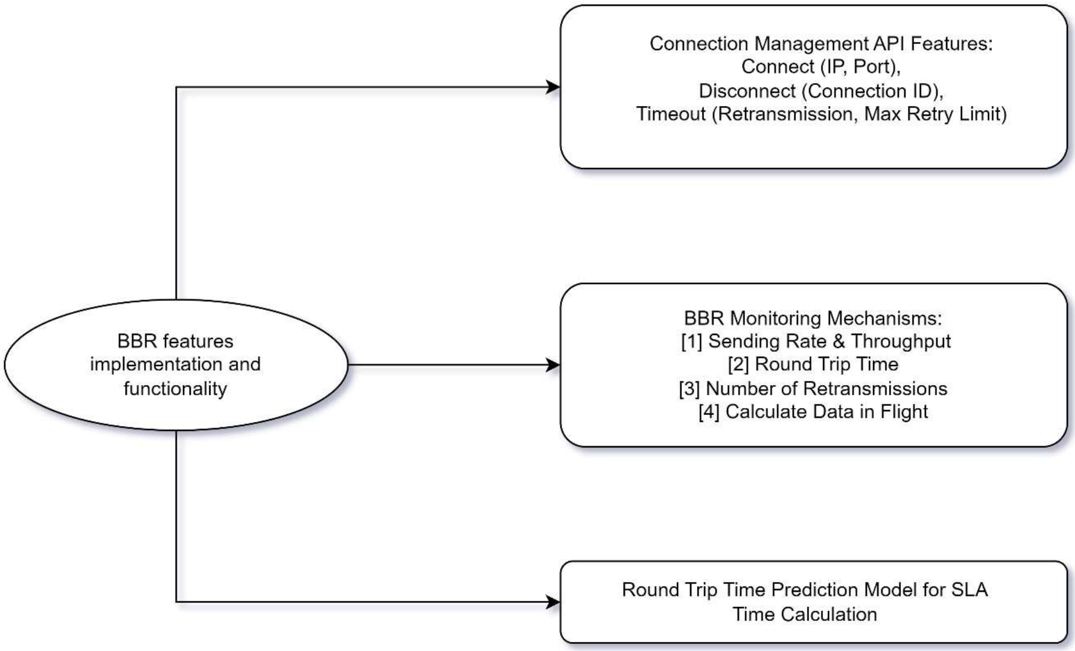
[3] **Choice and tradeoff between different processing models**: To be in order to fulfill the required SLA deadline, both server and client can choose and switch between different pretrained neural models with varying fine-grained result of image labelling versus their respective processing time. This aspect enables me to build a detailed model for decision making of computational offloading between edge client and server.

## First Sketch of the Idea:

Basic overview of the system would be as follows:


Switch between pretrained neural models in Python packages
Remote Cloud Server or Edge Server
Image Labelling Application + Recording:
[1] Timestamp of request from Edge Client
[2] Timestamp of response to Edge Client
BBR evalutaion and SLA calculation
Image Labelling Application + Recording:
[1] Timestamp of request to Remote Server
[2] Timestamp of response from Remote Server
Edge Client
Switch between pretrained neural models in Android libraries or SDK
Camera
UI + Controls

BBR with QUIC would be implemented as follows:


Connection Management API Features:
Connect (IP, Port),
Disconnect (Connection ID),
Timeout (Retransmission, Max Retry Limit)
BBR features implementation and functionality
BBR Monitoring Mechanisms:
[1] Sending Rate & Throughput
[2] Round Trip Time
[3] Number of Retransmissions
[4] Calculate Data in Flight
Round Trip Time Prediction Model for SLA Time Calculation

## Initial Time Plan:

As per the initial research and discussions, I have the following initial plan:

| Activity | Time Allocation |
|---|---|
| Literature review | 6 weeks = 1.5 months |
| Initial design phase | 2 weeks = 0.5 months |
| Implementation and Testing | 6 weeks = 1.5 months |
| Results Acquisition and Evaluation | 2 weeks = 0.5 months |
| Report Writing | 6 weeks = 1.5 months |
| Final Revision and Correction | 2 weeks = 0.5 months |
| Extra time buffer | 2 weeks = 0.5 months |

It’s worth noting that the scope implementation is expected be very extensive, so it would require early head start before total completion of Literature Review phase. Some consecutive / adjacent phases are expected to overlap up on each other.

I am planning to use a Trello Board for planning and monitoring of tasks, which I can also share upon your request. Further refined discussion would be needed to define “Must have” and “Optional items”.

Core goal is to have good evaluation of few items done correctly with proper written representation, rather than a large list of unfinished task items at the end with incorrect or incomplete evaluation and inadequate written representation.

**Related Work:**

The initial idea for BBR protocol has been discussed in the following paper, which serves as the starting point for further research:

[1] Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H. and Jacobson, V., 2017. BBR: congestion-based congestion control. Communications of the ACM, 60(2), pp.58-66.

This paper serves for better understanding of BBR in terms of core mechanism and metrics:

[2] Scholz, D., Jaeger, B., Schwaighofer, L., Raumer, D., Geyer, F. and Carle, G., 2018, May. Towards a deeper understanding of TCP BBR congestion control. In 2018 IFIP networking conference (IFIP networking) and workshops (pp. 1-9). IEEE.

This paper provides survey of different implementation of QUIC with BBR and their respective comparisons:

[3] Hertelli, Salim, Benedikt Jaeger, and Johannes Zirngibl. "Comparison of Different QUIC Implementations." Network 7 (2022).

This paper presents evaluation of Google Pixel 6 phone with BBR v1 and v2 with kernel level implementation of TCP and its performance comparison with TCP CUBIC:

[4] Vargas, Santiago, et al. "Are mobiles ready for BBR?." Proceedings of the 22nd ACM Internet Measurement Conference. 2022.

This paper provides fine-grained analysis of how to get better performance with tuning BBR and also its limitations:

[5] Cao, Yi, et al. "When to use and when not to use BBR: An empirical analysis and evaluation study." Proceedings of the Internet Measurement Conference. 2019.

**References:**

[1] Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H. and Jacobson, V., 2017. BBR: congestion-based congestion control. Communications of the ACM, 60(2), pp.58-66.

[2] Scholz, D., Jaeger, B., Schwaighofer, L., Raumer, D., Geyer, F. and Carle, G., 2018, May. Towards a deeper understanding of TCP BBR congestion control. In 2018 IFIP networking conference (IFIP networking) and workshops (pp. 1-9). IEEE.

[3] C. A. Grazia, N. Patriciello, M. Klapez and M. Casoni, "BBR+: improving TCP BBR Performance over WLAN," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9149220.

[4] [Last accessed 21.01.2023] https://cloud.google.com/blog/products/networking/tcp-bbr-congestion-control-comes-to-gcp-your-internet-just-got-faster

[5] Cardwell, N., Cheng, Y., Yeganeh, S.H., Swett, I., Vasiliev, V., Jha, P., Seung, Y., Mathis, M. and Jacobson, V., 2019, March. Bbrv2: A model-based congestion control. In Presentation in ICCRG at IETF 104th meeting.

[6] Y. -J. Song, G. -H. Kim, I. Mahmud, W. -K. Seo and Y. -Z. Cho, "Understanding of BBRv2: Evaluation and Comparison With BBRv1 Congestion Control Algorithm," in IEEE Access, vol. 9, pp. 37131-37145, 2021, doi: 10.1109/ACCESS.2021.3061696.

[7] Abbasloo, Soheil, et al. "Toward Optimal Performance with Network Assisted {TCP} at Mobile Edge." *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*. 2019.

[8] Fiandrino, Claudio, et al. "openLEON: An end-to-end emulation platform from the edge data center to the mobile user." *Computer Communications* 148 (2019): 17-26.

[9] [Last accessed 24.01.2023] https://github.com/google/bbr/tree/v2alpha

[10] Vargas, Santiago, et al. "Are mobiles ready for BBR?." Proceedings of the 22nd ACM Internet Measurement Conference. 2022.

[11] Hertelli, Salim, Benedikt Jaeger, and Johannes Zirngibl. "Comparison of Different QUIC Implementations." Network 7 (2022).

[12] Cao, Yi, et al. "When to use and when not to use BBR: An empirical analysis and evaluation study." Proceedings of the Internet Measurement Conference. 2019.