# A Comparison of Streaming Sensory Data Transfer Methods in Remote Processing Environment

**Arush Nagpal[1], Prashant Singh Rana[2]**

[1]Thapar University, Patiala, India

[2]Assistant Professor, CSED, Thapar University, Patiala, India

**Abstract:** ***The advancements in Internet of Things (IOT) has resulted in ubiquitous availability and use of sensors in remote single board computers like raspberry pi or IOT device specific hardware. Though every device has a local storage and processing component but it is not fast and scalable enough to process the huge amount of generated by the sensors involved. The data has to be offloaded to different large computing and processing systems using network. This data can further be used for monitoring, prediction or analysis. Even though there are different communication protocols and architectural styles, namely REST, SOAP, socket and RPCs to efficiently transfer this data but no concrete quantitative data is present to show the efficiency of these methods in different environments. This paper describes these numerous telemetry approaches available to transfer data to a remote server and presents different configurations where each method can prove to be the most effective with respect to different type of sensor data. It also provides a concrete analysis on the type of communication mechanism optimum for a particular length of data transmission.***

## 1. Introduction

Each IOT device is bundled with data acquisition, data processing and data communications module along with the hardware sensors. The most up-to-date devices (tiny computers like raspberry PI or mobile devices running on Android, IOS etc.) provide users with even better resources with a higher capacity: computing modules(single core/multi core processors), storage capacity (RAM/storage disks) and communication modules along with an OS. In spite of such hardware, there are a lot of practical limitations on the processing of data that can be done locally on the device. Enabling support to run complex computations on such devices like machine learning algorithms (SVMs, neural networks) or data analytics algorithms (map reduce) would make the device very expensive and heavy. A natural way to overcome this limitation is to stream this data to remote processing systems. Much work has been done to compare the energy efficiency and use cases comparing one protocol to another in use cases but no quantitative data is available which can help choose the right approach and illustrate the percentage improvements in speed for streaming different data lengths.

This paper is organized as follows. Section II describes the type of sensors and the data involved. Section III and IV discuss the communication protocols available and the protocols used for our experiment. Section V describes the method used for the experiment and the section VI sheds light on results further followed by the conclusion.

## 2. Sensors and Data Involved in Different Mobile Devices

There are various types of sensors that can be attached to a remote device or are already built in. In this paper, we have discussed five types of sensors and their output data. The techniques used can also be extended similarly for not only other sensors but also any type of data required to be sent over network. Any kind of sensor will emit raw data which might have to be converted to a different form for its effective use. This raw data is usually in the form of byte buffer and is directly obtained from the small hardware buffer associated with the sensor. For example, a microphone will output all the recording into a fixed length hardware buffer memory which can be extracted for processing into the device memory using a computer program.

The raw data is extracted continuously after fixed interval of time. This interval varies is decided by the sampling rate of sensor. Sampling is the reduction of a continuous time signal to a discrete time signal. Sampling rate is the average number of samples obtained in one second. Table 1 classifies the sensors studied along with the significance of raw data generated by them.

**Table 1:** Types of sensors and their outputs

| Sensor | Raw data output | Significance of the output |
|---|---|---|
| Proximity Sensor | A single float value array | The array specifies the distance between the sensor and a nearby object in centimeters |
| Gyroscope Sensor | An array of three float values | Each value in the array specifies the angular velocity of the device along X, Y and Z axis in radians/second |
| GPS Sensor | A typical $GPGGA message is an array of 16 values | The array has values of different GPS parameters like latitude, longitude, quality, checksum, altitude etc. |
| Microphone sensor | A byte array of variable length based on the audio sampling rate | The unique values in array is actually the speech data encoded in a pcm raw data format |
| Camera sensor | Three 2-D arrays of numbers ranging from 0 to 255 specifying intensity of Red, green and blue colors | A combination of all values in the three arrays creates a unique color in each pixel of the image. The array size is determined by the camera resolution. |

## 3. Transmission of Data from Device to A Remote Server

Each mobile device might accommodate more than one sensor. For long term post processing of the sensors' data, data is required to be streamed from the mobile device to a different server with large computation and storage capacity. But the raw data in array form cannot be directly transmitted over the network. It has to be encoded to a universal format so that it can be safely transferred over legacy channels avoiding data corruption. A very popular encoding mechanism known as Base64 encoding is used for our experiment. The decoding is done at the end of the server as soon as the data is received as shown in Figure 1.

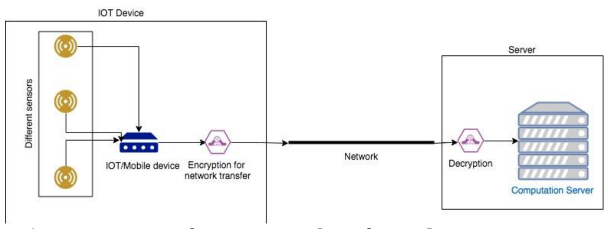

**Figure 1:** Transferring raw data from device to server

## 4. Encoding raw data output of sensor for transfer over the network

Each sensor has a different output format which, after encoding, can result in different lengths of data required to be transmitted. The length of the data to be transmitted has a direct correlation with the transfer speed.Hence, we try to convert the output to base64 encoded format and figure out the length of the encoded data.

For example, a proximity sensor output seems to be like [“23.47”]. Encoding it using Base64 gives an output like:
W+KAnDIzLjQ34oCdXQ=

Length: 21 bytes

**Table 2:** Data length after base64 encoding

| Sensor | Approximate output length after base64 encoding (bytes) |
|---|---|
| Proximity Sensor | 21 |
| Gyroscope Sensor | 56 |
| GPS Sensor | 200 |
| Microphone sensor | 5500 |
| Camera sensor (128x128 image) | 1440000 |

## 5. Method

There are different ways to transfer data over the network and we decided to evaluate the speed and efficiency of different communication protocols and architectural styles, namely REST, SOAP, Socket and gRPC. gRPC (a specific implementation of RPCs) is chosen due to its efficient serialization/deserialization and industry accepted standard for RPC calls. All the experiments were done with a standard MTU (maximum Transmission Unit) of 1500. These methods were chosen because they are well established ways of interoperating data between systems.

We chose an android mobile device for capturing and transmitting data from the sensors. The device used is a Google Pixel with 1.6GHz quad-core processor and 4 Gb of RAM and the android version Android 7.1.The transmission time is considered only for a unidirectional data flow from the device to the server because the use case considered is only to stream the data to the server and not from it.

Data of different lengths was transmitted and the times in milliseconds was recorded both before sending the data (from the mobile device) and after receiving the data (on the server). Both the devices were put on a dedicated network with no other data transmission happening over the network in order to count only the transmission delay and not the routing delay. Each length of data was transmitted 30 times and an average of the transmission times was taken to count for the occasional network delay. The lengths of transmission data taken included sensor data as well as different data lengths so as to include all lengths of data.

## 6. Results

The results obtained have been split into two graphs according to the size of data due to the behavior depicted by the transmission times. Figure 2 shows

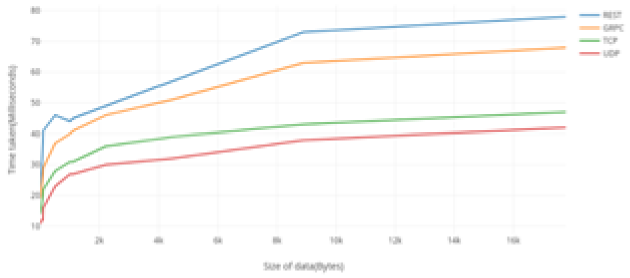

**Figure 2:** Time taken for data transmission for data upto 17700 bytes (~17.29kB)

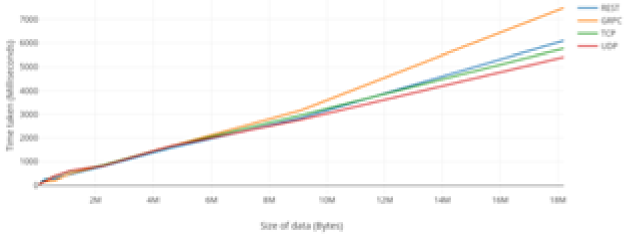

**Figure 3:** Time taken for data transmission for data beyond 17700 bytes (~17.29kB)

For small data (less than 18 kB), UDP and TCP seem to be much faster than REST or gRPC. UDP seems to be much fastertillarounddatasizeof90000bytes(9kB) but as the size increases UDP transfer time becomes comparable to TCP[6]. It is because any TCP library handles large files automatically (splits into multiple packets) but for UDP, it needs to be handled manually. REST takes longer time than all the other three mechanisms and as the size increases, REST becomes even moreslow. Data transfer using gRPC is consistently faster than REST. The same trend continues till a size of 1Mb, after which gRPC slows down. It is due to the fact that to support gRPC to transfer more than 1 Mb in a single shot, 2 packets have to be formed manually and sent over the network.

UDP might seem to be the fastest but it carries along many limitations. UDP does not guarantee all the data to reach the server. Even if one packet is dropped due to network latency, it leaves it and continues to send the nextpacket.

ATCP socket is the next best choice which offers great speed but only when the data is of the same form. For example, ifa device has more than 2 sensors and the data from all the sensors has to be sent, TCP will have to open multiple connections to send the packets. gRPC, on the other hand is optimized to send different data in a single go, optimized on the data structure [4]. gRPC is the best choice when the data contract is known between the two parties and there are different types of data involved. Instead of opening different connections. gRPC can bundle all data in a single packet in binary format and stream it faster than any other type of communication[5].

Receiving architecture- Based on the type and sampling rate, the mobile device and the server can be configured in 4 different ways to receive all the data:

a) Stream direct process direct (SDPD) – Data is streamed to the server directly as soon as it is captured by the mobile device from the sensor. The server operates on the datadirectly.
b) Stream direct process in batch (SDPB) – Data is streamed to the server directly but for processing, theserverkeepsabufferandkeepsonappendingthe data into the buffer until a fixed size is reached and then startsprocessing.
c) Stream in batch process direct (SBPD) – Data is accumulated at the mobile device for some timeand while the server processes it directly. In this case, the size of data transmission increases and hence, takes longertime.
d) Stream in batch process in batch (SBPB) – Data is accumulated before sending as well as while processing. The size of data transmission increases and it takes a longertime.

To minimize the transfer time, architectures SDPD and SDPB should be chosen over SBPD or SBPB because of the speed at which small packets can be transferred over the network. Instead of spending memory and computation capacity on accumulating data on mobile device, the accumulation process can be done on server side resulting in fast communication.

*Limitations.*A lot of challenges were faced during transfer of such data using the mentioned protocols. Each method has a maximum packet size which can be sent over the network. The maximum size of a TCP packet is 65535 bytes. The same is 65507 bytes for UDP. But due to MTU being fixed at 1500, the effective size is a lot less. gRPC has a maximum message size of 4 MB. Thus, to calculate the time taken for data transfer was calculated by programmatically splitting the data into chunks which were then transferred using the above protocols. This reduces the effective speed transfer as it involves Therefore, it is recommendedto keep the payload size as least as possible.

*Effect of sampling rate on data transfer*. The rate of data transfer is limited by the rate at which data is being generated by the sensors. For example, if microphone sensor is generating a data of 5500 bytes 20 times in a second, then the transfer mechanism should take less than 1/20 seconds to transfer 5500 bytes over the network.

## 7. Conclusion

The experiment gave a comparative analysis on the efficiency and speed of different data communication mechanisms. We were able to discard 2 specific architectures, namely SBPD and SBPB, of sensory data transfer from mobile devices for efficient data transfer. Each method has its own specific range of packet size where it works best. We were able to set up a convention on when to use either of the protocols. REST is not recommended in any case. If the loss of data from any sensor is not important and can be ignored, UDP is the fastest and most efficient approach. UDP is preferred if the server is local to the mobile device (same network). TCP should be used if each packet is important. In case additional data needs to be sent apart from the base64 encoded string data from sensors, gRPC is the bestchoice. An effective way to compare data transfer speeds for all the methods was laid forward. Such results enable us to verify if a protocol is suitable for data transfer of sensors generating X bytes with the frequency of Y times per second.

## References

[1] Chamas, C. L. (2017). Comparing REST, SOAP, Socket and gRPC in computation offloading of mobile applications: An energy cost analysis. *2017 IEEE 9th Latin-American Conference on Communications* .Latincom.
[2] M. Halpern, Y. Z. (2016). Mobile CPU's rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction. *High PerformanceComputer Architecture (HPCA)* (pp. 64-76). 2016 IEEE International Symposium on. IEEE.
[3] Steven Han, R. L. (2016). *Teaching the Internet of Things: Bridging a Path from CPE329.*
[4] Wang, X. Z. (1993). GRPC: A Communication Cooperation Mechanism in Distributed Systems. *ACM SIGOPS Operating Systems Review.*ACM.
[5] Carreno, E. D. (2017). IoT Workload Distribution Impact Between Edge and Cloud Computing in a Smart Grid Application. *High Performance Computing: 4th Latin American Conference.* CARLA2017.
[6] Hofmann, P. C. (2007). Analysis of UDP, TCP and voice performance in IEEE 802.11 b multihop networks. *13th European Wireless Conference.* IEEE.
[7] K. Kumar, J. L.-H. (2013). *A survey of computation offloading for mobilesystems.*
[8] T. Salah, M. J.-Q. (2016). "The evolution of distributed systems towards microservices architecture. *Internet Technology and Secured Transactions (ICITST.* IEEE).