San Jose State University

Project Report on,

Adaptive Streaming

Under the guidance of Prof. John Gash
CMPE 275 (Spring 2022)

Prepared By
Dhrupa Patel (015265887)
Sai Kiran Madupu (015505243)

Goal

Developing an Adaptive Algorithm that manages changes in bandwidth in real-time. This lab will focus on conditions other than failure detection like variable bandwidth, packet failures, spotty connections, etc.

Approach

- In this research work we aim to develop an algorithm in which the packet size of the next packet adapts to the changing bandwidth of the channel.
- Various experiments were carried out for different packet sizes and variable bandwidth to know its effects on network performance.
- To ensure correct data is transmitted through the channel, hashing technique has been employed to encode and decode the digest of data transmitted through the channel and later verified if data has been correctly sent.

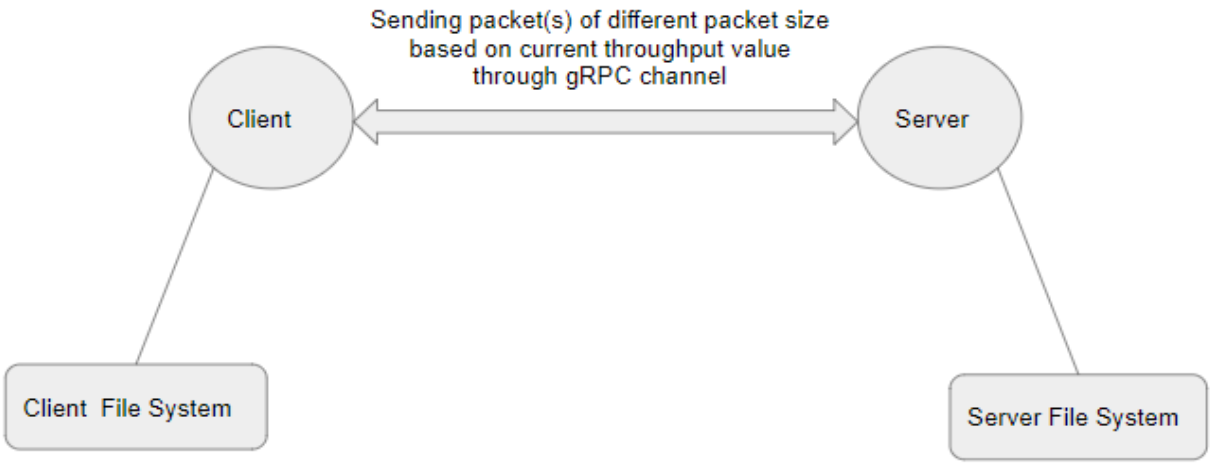
Communication Techniques:

gRPC Channel

Like many RPC systems, gRPC is based around the idea of defining a service, specifying the methods that can be called remotely with their parameters and return types. By default, gRPC uses protocol buffers as the Interface Definition Language (IDL) for describing both the service interface and the structure of the payload messages. gRPC allows us to define four kinds of service methods namely, unary RPCs, Server streaming RPCs, Client streaming RPCs, Bidirectional streaming RPCs. In this lab we have used unary RPC service. Unary RPCs where the client sends a single request to the server and gets a single response back, just like a normal function call.

Choice of I/O function

A data file from the client is sent to the server using the grpc channel. An adaptive algorithm will keep sensing the channel for it's throughput every 10s and update the size of the next packet that should be transferred through the channel. Client and Server will be on the same network connection. To ensure that data sent is received correctly on the server end, a hash digest of the packet is generated and sent through the channel and verified on the other end.



Experiments Conducted:

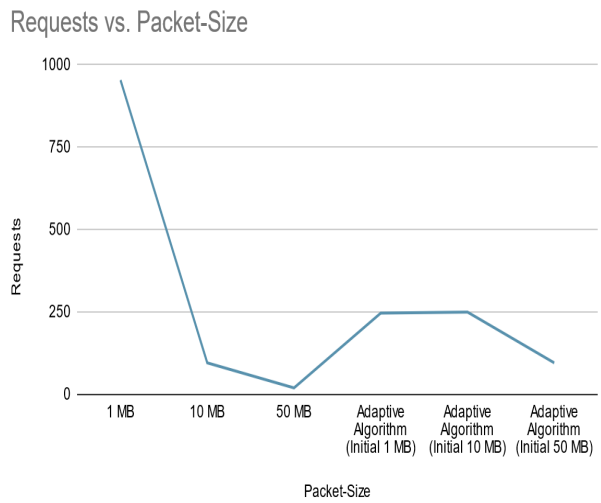
Network Conditions:
Constant Bandwidth: 50 Mbps
Variable Bandwidth: varying b/w pattern is as follows:

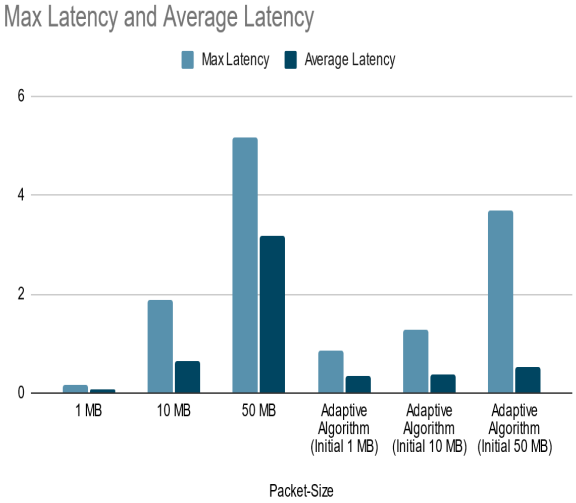
0-30 secs: max bandwidth (50 Mbps)
30-60 secs: 6.25 Mbps with 5% packet drop
60-90 secs: 18.75 Mbps with 2% packet drop
90-xx secs: max bandwidth (50 Mbps)
Speed is in megabytes per second.

Bandwidth	Adaptive Algorithm	No. of Clients	File Size (GB)	Packet Size (mB)	Requests	Time taken (secs)	Max Latency (secs)	Avg Latency (secs)	Memory (kB)	CPU Utilization (%)	Initial Packet size (mB)	Avg Packet Size (mB)
Constant	OFF	1	1	1	954	60.16	0.15	0.07	22,320	48	1	1
Constant	OFF	1	1	10	96	64.53	1.89	0.66	22,944	47.2	10	10
Constant	OFF	1	1	50	20	64.19	5.16	3.18	22,464	47.3	50	50
Constant	ON	1	1	1	247	87.38	0.87	0.35	22,192	47.6	1	3.78
Constant	ON	1	1	10	250	99.22	1.28	0.39	22,544	37.5	10	3.82
Constant	ON	1	1	50	96	83.22	3.7	0.54	22,640	43.4	50	6.28
Varying	OFF	1	1	1	954	130.2	3.09	0.14	22,272	27.6	1	1
Varying	OFF	1	1	10	96	135.36	20.79	1.4	22,448	26.9	10	10
Varying	OFF	1	1	50	20	142.91	38.56	7.11	22,192	19.4	50	50
Varying	ON	1	1	1	182	150.95	20.37	0.82	22,256	21.6	1	5.27
Varying	ON	1	1	10	161	144.24	12.34	0.89	22,992	22.6	10	5.94
Varying	ON	1	1	50	136	163.86	19.77	1.2	22,544	21.6	50	7.02

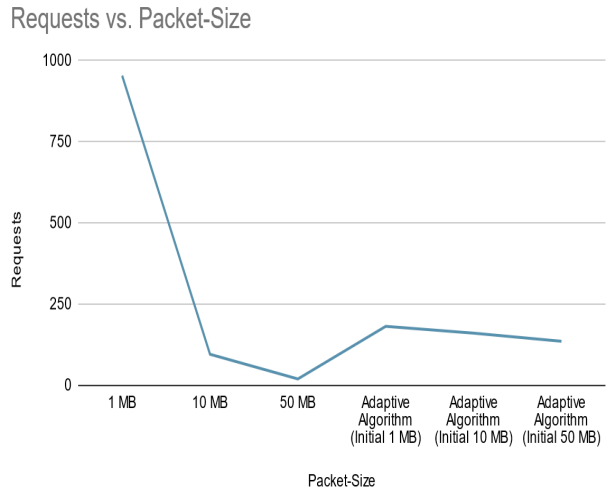
Visual Representation of data:

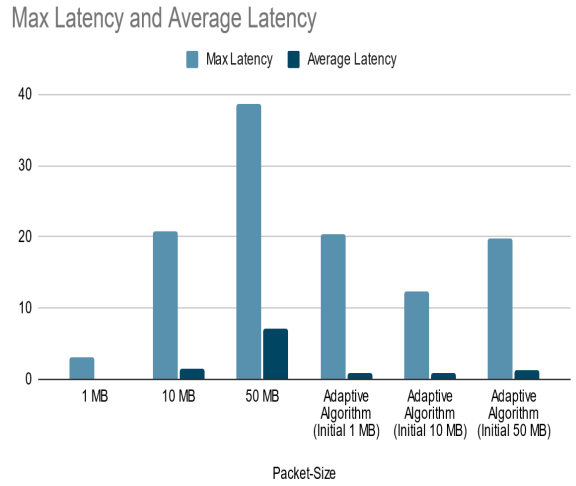
Constant Bandwidth:





Varying Bandwidth:





From the above data it's quite evident that the developed adaptive algorithm works and improves the latency of the network which is one of the important factors while streaming the data. Above algorithm task is to decide the tradeoff between number of requests and latency.

Performing experiment with two clients and one server:

Adaptive Algorithm	No. of Clients	File Size (GB)	Packet Size (mB)	Requests	Time taken (secs)	Max Latency (secs)	Avg Latency (secs)	Memory (kB)	CPU Utilization (%)
OFF	2	1 (Client-1)	10	96	171.54	16.21	1.77	23380	7.1
		1 (Client-2)	10	96	241.17	22.02	2.5	22544	4.1
ON	2	1 (Client-1)	10	348	197.02	7.97	0.56	23268	7
		1 (Client-2)	10	333	241.17	27.16	0.72	22656	3.5

Above statistics provide clear understanding that adaptive algorithms will work efficiently with varying bandwidths of the channel as even if there is an increase in clients the latency is not getting affected. Thus, it can be concluded that packet size is adapting to varying bandwidth of the network and making transmission more reliable and efficient.

Conclusions:

- Making a channel adaptive to changes in bandwidth in real time to reduce the overhead on the network and possibly reduce the number of packet loss (for UDP Connections).
- Bandwidth will be utilized at its best, if packet size has a linear relationship with changing bandwidth of the channel.

Future Enhancements:

- For this research lab we have made certain conditions fixed like if there is an addition of ~2000 bytes in throughput it means that there is traffic on the network and the next packet should be re-adjusted to avoid congestion or packet loss. Thus, more experiments can be performed on what difference value should be opted to trigger the change in the next packet size.
- For carrying out current experiments we have set the next packet size to the half of throughput value (actual bandwidth). Research can be performed to get the best next packet size for current throughput value to utilize bandwidth at its best.

References:

- https://psutil.readthedocs.io/en/latest/
- https://www.tutorialspoint.com/grpc/grpc_unary.htm
- https://grpc.io/docs/what-is-grpc/core-concepts/
- https://www.geeksforgeeks.org/md5-hash-python/
- https://cseweb.ucsd.edu/classes/sp01/cse126/

Contributions:

Dhrupa Patel: I was responsible for coding adaptive layer parts which sense the medium at fixed intervals and update the size of packet to be sent next based on throughput value. Apart from this I have also conducted few tests on the algorithm using different testing conditions.

Sai Kiran Madupu: I was responsible for establishing a gRPC channel between client and server and also checking if the data received and sent are the same using the md5 hashing algorithm. Apart from this I have also conducted tests on the algorithm using different testing conditions.