EVALUATION OF EDGE CLOUD SERVICE SCENARIOS WITH APPLICATION SPECIFIC ROUTING IN THE NETWORK

BY ANUSHA SHEELAVANT

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Dr. Dipankar Raychaudhuri

and approved by

_____

_____

_____

New Brunswick, New Jersey

October, 2015

ABSTRACT OF THE THESIS

Evaluation of Edge Cloud Service Scenarios with Application Specific Routing in the Network

by Anusha Sheelavant

Thesis Director: Dr. Dipankar Raychaudhuri

This thesis presents the evaluation of edge cloud service scenarios with Application Specific Routing (ASR) in the network. Edge cloud service is a computing service provided by decentralized servers which reside at the edge of the network and provide services to client devices in the local area. By being in close proximity to end users, there is a reduction in latency that may provide improved Quality of Experience (QoE). The edge cloud provides services to Internet of Things (IoT) or Cyber Physical Systems (CPS), such as, mobile phones, sensors, vehicles and augmented reality using devices like Google glass, which host real time applications and require quick response. This work evaluates the performance of such edge cloud services for mobile users in the MobilityFirst (MF) architectural framework.

The Hadoop open source cloud software framework has been used in this study for parallel and distributed processing of jobs at the edge cloud site. The performance of Hadoop has been benchmarked for different network scenarios by varying the link quality: bandwidth and latency. Results show that both link parameters and edge cloud performance have a significant impact on the response time for a client's cloud service request. Hence, this work uses both these metrics in making routing decisions to improve end users QoE with help of Virtual Network (VN) capabilities. VN on

MF enables specification of customized network topologies and routing algorithms for specific application scenarios such as cloud service. ASR is one such routing algorithm that makes path selection based on application specific parameters that affect end users' QoE such as the link and edge cloud performance metrics. The link performance metric incorporates bandwidth and latency in the link from client to the edge cloud. The edge cloud performance metric on the other hand captures the average job completion time at the edge cloud site which gives a sense of the workload, the number of nodes and the processing capability of the cluster. With the aid of ASR on MF VN, the best edge cloud site for a client's request is chosen with the objective to minimizing the overall response time.

Evaluation of users' QoE for edge cloud services was performed on the ORBIT testbed by replicating cloud services on multiple edge cloud sites. Workload was induced at each of these sites to control the job completion time and link quality is varied to affect the link performance. Response time for a client's request is evaluated using ASR and also using the basic anycast service in MF, for performance comparison. While anycast always chooses the nearest edge cloud, ASR provides the best response time by either choosing a lightly loaded cluster or by choosing a high quality link or a combination of both, and therefore provides a better QoE to the end user.

Acknowledgements

I would like to express my sincere gratitude to my advisor Professor Dipankar Raychaudhuri for his continual support and encouragement. It has been a great learning experience and a pleasure working under his supervision. I am also grateful to my committee members Professor Wade Trappe and Professor Janne Lindqvist for their valuable suggestions.

I would like to thank Ivan Seskar for always being there to give feedbacks and answer every doubt I approached him with. I am also thankful to Professor Yanyong Zhang for her guidance during the initial stages of this research work. I would also like to acknowledge Francesco Bronzino, Aishwarya Babu and Kai Su, members of the MobilityFirst group at WINLAB, for all the invaluable discussions we have had through out the course of this research work.

I am eternally grateful to my family for their love, support, and unwavering belief in me. Lastly, I would like to thank my friends at WINLAB and Rutgers for making this journey, a pleasant one.

Table of Contents

List of Figures

Chapter 1

Introduction

The rise in popularity of smart phones has increased interest in the area of mobile Cyber Physical Systems (CPS). Smart phone platforms make ideal mobile CPS for a number of reasons, like: availability of significant computational resources (such as processing capability and local storage) and multiple sensory input/output devices (such as touch screens, cameras, GPS chips, light sensors, proximity sensor, etc.).

For tasks that require more resources than are locally available, one common mechanism is to link the mobile system with either a server or a cloud environment, enabling complex processing tasks that are time consuming to process under local resource constraints. Since CPS is associated with real time applications, the cloud and networking environment must be reliable and must be able to provide quick response upon service requests. Thus, cloud computing services are being pushed to the edge of the network to provide quick response, resulting in a scenario called edge cloud or fog computing [1, 2] as edge cloud services.

1.1 Problem Statement

Edge cloud service is a computing service provided by decentralized servers which reside at the edge of the network and provide services to client devices in the local area. By being in close proximity to the mobile end users, there is a reduction in latency that may provide improved Quality of Experience (QoE). Edge cloud provides services to Internet of Things (IoT) or Cyber Physical Systems, such as, mobile phones, sensors, vehicles and augmented reality using devices like Google glass, which host real time applications and require quick response [2].

The MobilityFirst (MF) [3], Future Internet Architecture(FIA) [4] proposes a new

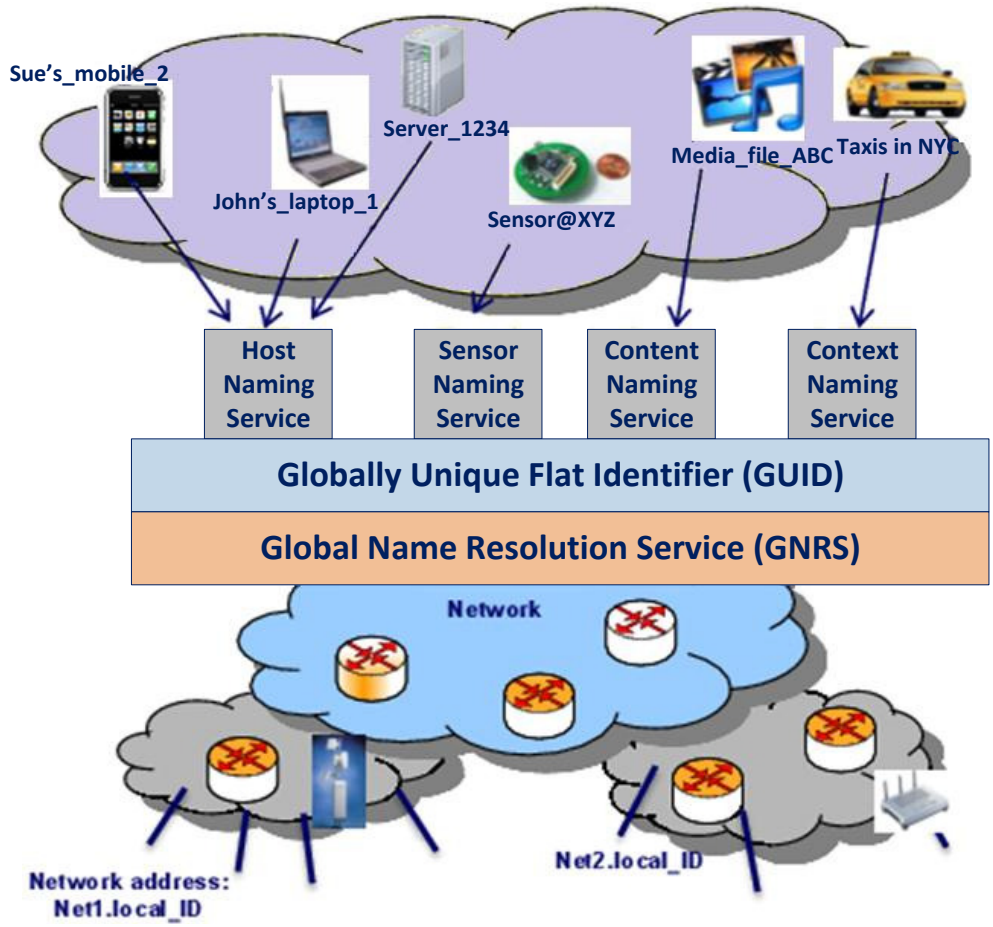

Figure 1.1: Key design features of MobilityFirst.

named-object networking paradigm that is well suited for handling mobile nodes and edge cloud service scenarios. As outlined in figure 1.1, in MobilityFirst, every object or service - mobile phone, an embedded sensor, cloud site, or a cloud service is assigned a Globally Unique Identifier (GUID) which is independent of its current point of attachment to the network.

The Global Name Resolution Service (GNRS) [5], a key feature of MobilityFirst, provides a dynamic binding between GUID's and their current network locators. This makes it possible to support seamless mobility and service replication in a very natural way, thus providing a strong foundation for building advanced cloud services across global network. Since MF takes care of mobile objects and supports seamless mobility, the remaining challenge we face here is to provide satisfactory quality-of-experience

(QoE) to mobile users for edge cloud services.

Edge cloud services reside at the edge of network and are in close proximity with the clients, reducing latency to some extent [6]. This may improve user QoE by reducing the response time for a user's request. Response time is the time that a user has to wait for a response to a request that has been made. This work evaluates the performance of such edge cloud services for mobile users in the MobilityFirst architectural framework.

In this work, we focus on achieving improved QoE for a user by investigating the factors that affect the response time for a client's request and their contributions to the response time. The Hadoop open source cloud software framework has been used in this study for parallel and distributed processing of jobs at the edge cloud site. We benchmark the performance of Hadoop for different network scenarios by varying the link quality: bandwidth and latency, and performance of the edge cloud. The effect of these parameters on response time is studied.

This work uses both link parameters and edge cloud performance metric in making routing decisions to improve end users QoE with help of Virtual Network (VN) capabilities. VN on MobilityFirst (MF) enables specification of customized network topologies and routing algorithms for specific application scenarios such as cloud service. ASR is one such routing algorithm that makes path selection based on application specific parameters, such as the link and edge cloud performance metrics, that affect the end users' QoE . The link performance metric incorporates bandwidth and latency in the link from client to the edge cloud. The edge cloud performance metric on the other hand captures the average job completion time at the edge cloud site which gives a sense of the workload, the number of nodes and the processing capability of the cluster. With the aid of ASR on MF VN, the best edge cloud site for a client's request is chosen with the objective to minimizing the overall response time.

1.2 Related Work

Traditional networks forward packets from source to destination using the shortest path but with innovations in networking, such as Software Defined Networking (SDN),

networks are forced to be application aware to improve user's QoE. The key to improving user's QoE is providing visibility of application parameters at the network level in order to enable improved routing decisions. Application aware routing has been proposed for SDN where an SDN controller monitors the network conditions and ensures user's QoE [7].

In our work, we use this key concept of introducing application awareness for improving user's QoE by proposing ASR which is supported by VN on MF. We delve into the factors that affect user's QoE and study their contributions on response time. Further, we derive a decision metric for ASR, which consists of parameters that affect the response time and use this decision metric for making routing decisions to ensure user's QoE.

Improving user's QoE has been one of the important topics of interest in the recent times. In order to reduce the latency between a mobile client and the cloud environment, service migration is done when a mobile client connects to a new edge cloud and if the client's data is in the old edge cloud. Service migration is the process of transferring data from one edge cloud site to another. Once the migration is complete, processing of client's data resumes at the new edge cloud.

There are many approaches to service migration. One approach is called as VM migration, in which an executing VM is suspended, it's processor, disk and memory state are transferred, and then the VM execution begins from the point of suspension at the new edge cloud site. In the second approach, known as dynamic migration, all the edge cloud sites have a base VM, and only the overlay is transferred. The base VM is combined with the overlay and is known as launch VM [6]. The major contributions to generating a launch VM are overlay transmission and decompressing/applying the overlay on the edge cloud. Service migration must be done only if the new edge cloud site can improve the user's QoE as compared to the old edge cloud since transmission costs incurred during migration are quite high, and could defeat the whole point of service migration. Hence, in our work, with the ASR algorithm, the application running on the edge cloud could decide if service migration is necessary, and only then migrate. This could reduce unwanted transmission costs and provide the user with better QoE.

Chapter 2

Overview of MobilityFirst

MobilityFirst [8] is a clean-slate redesign of the Internet architecture which supports dynamic mobility at scale by cleanly separating names or identifiers from addresses or network locations. It is centered on a new name based service layer which uses the concept of flat GUIDs for network attached objects, a single abstraction that covers a broad range of communicating objects from a simple device such as a smartphone, a person, a group of devices/people, content or even context. GUIDs are basically public keys assigned by a name certification service to the networks objects and are the long-lasting network level identifiers for these objects. Network services invoked on messages are defined first and foremost by the source and destination GUIDs.

For routing, a hybrid name/address based scheme is used for scalability, employing a fast Global Name Resolution Service (GNRS) to dynamically bind the destination GUID to a current set of network addresses (NA). The GNRS thus forms a central feature of the mobility-centric architecture, enabling on-the-fly binding of names to routing addresses as needed for dynamic mobility, disconnection or cloud migration scenarios.

2.1 Service GUID

In MobilityFirst, the permanent host-identifiers are GUIDs, which are dynamically mapped to the network addresses (NAs) through a centralized mapping service called the Global Name Resolution Service (GNRS). Every application that runs on the servers/ cloud site has it own service GUID and there could be multiple cloud sites that host the same application with one service GUID. When a request is made for a particular service GUID, any cloud site hosting that application is eligible to service

the request. With the help of VN on MF, ASR helps in routing the request to the best edge cloud site that can provide good overall response time. This in turn aids in balancing the application traffic among different cloud sites to make the best use of available resources. In other words, ASR helps in load balancing and makes the best use of already available resources to ensure user's QoE.

2.2 GSTAR Routing

In GSTAR [9], all nodes periodically broadcast fine grained link quality information, in the form of flooded link state advertisements(F-LSAs), that contain the short term and long term Estimated Transmission Time(SETT and LETT respectively) of their current 1-hop neighbors. These are then used to calculate the overall path quality to all other nodes using Dijkstra's shortest path algorithm . Path selection and transmission decisions are based on factors such as link availability and link quality in terms of the ratio of SETT over LETT. All nodes periodically probe for neighbors, making a note of which neighbors are currently available and what the ETT (directly computed) for the links are. Over a sufficiently long period of time, they average the ETT values for a single link and compute a "long term ETT" value. In this work, the ETT values are a function of both bandwidth and latency, similar to the cost metric calculation [10] for EIGRP routing protocol [11]. The ETT values are given by:

$$ETT = 256 * (\frac{10^7}{BW}) + \sum_{i=1}^{n} \text{delay}(i) \tag{2.1}$$

Where BW is the bandwidth in $Kbps$ and delay(i) is the delay in 10's of μs on every link along the path.

We use these aggregated SETT values to measure the network performance which contributes to the ASR metric. Aggregated SETT is the sum of all the ETT values along the path from source to destination which is more than one hop away and is available in all the virtual routers on the VN.

2.3 MobilityFirst Virtual Network

MF supports virtual networks [12] which are logical abstractions of the physical router substrate that runs MF stack. Since every object in the MF network has a GUID, an application, service or even a whole virtual network can be identified by a GUID. The GNRS keeps track of various mappings, the VN's GUID to its member virtual router GUIDs and each virtual router GUID to its true GUID.

The physical substrate uses the GSTAR routing protocol to move packets across the network. The GSTAR routing tables are populated based on the link state messages that are exchanged among the router nodes. This routing table information is made available to the virtual layer in order to compute link performance metric. The edge cloud also advertises its performance metric using Application Specific Packets (ASP). ASR runs at the virtual layer to combine both link and edge cloud performance metrics to make routing decisions.

Chapter 3

Cloud Computing Framework

3.1 Hadoop

The system model is designed to have Hadoop software framework [13] running on edge cloud sites to process client requests. Hadoop is an open source software framework which is used for distributed storage and large scale processing of data sets on computer clusters. There are two components to Hadoop :

- Hadoop Distributed File System (HDFS)

  HDFS [14] is the primary distributed storage used by Hadoop applications which is based on Google's Google File System (GFS). The files are stored as blocks or chunks, usually of size 64MB. HDFS cluster consists of two parts: NameNode and DataNode. NameNode manages the file system metadata while DataNodes store the actual data. Data is replicated across several DataNodes to provide reliability and to overcome node failure. When a job is submitted to the cluster, HDFS breaks the job into chunks and distributes it to the nodes in the cluster. The NameNode and Datanodes have built in web servers which make it possible to check current status of the cluster.

- MapReduce

  Hadoop MapReduce [15] is a software framework for processing large amounts of data in parallel on clusters in a reliable, fault tolerant manner. A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file system. The framework takes care

of scheduling tasks, monitoring them and re-executes the failed tasks. Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes.

The MapReduce programming model essentially consists of three phases:

1. Map: This phase performs parallel tasks on a given data. The input data on which the processing is supposed to be done, is split across multiple servers in the cluster and is the input to the mappers. The exact task to be executed inside Map function needs to be provided by the programmer. Since the Map phase executes the same task on all input splits, the programmer needs to make sure that the functioning of the Map function does not depend on input data. Map phase is completely parallel, i.e. all input splits are processed in parallel, hence it provides faster results. The output of the Map phase is written to a temporary file in terms of key-value pairs. With the Hadoop framework, the only responsibility that the programmer has to carry out is writing these Map and Reduce functions. Hadoop framework, using its cluster management services, decides which server to put the task on to.

2. Shuffle: After Map phase generates its output in terms of key-value pairs, shuffle-and-sort phase sorts all key-value pairs generated by all map tasks, based on keys. After the keys are sorted, similar keys are put in the same bucket and forwarded to Reduce functions on the nodes that run reduce tasks.

3. Reduce: This is the last phase of MapReduce programming model. In this phase, all the key-value pairs are combined to get the final result. Similar to Map function, Reduce function is written by the application programmer. At the end of the reduce phase, the output (key-value pairs) is written back to HDFS.

3.2 Scheduling in Hadoop

The performance of Hadoop is dependent on its scheduler, which reactively assigns tasks to servers for computation using a particular scheduling algorithm. Hadoop has different scheduling algorithms, such as first in first out (FIFO) scheduling, fair scheduling and capacity scheduling. The Hadoop job scheduler for HDFS and MapReduce is shown in figure 3.1.

- HDFS : As the jobs are submitted to the cluster, they are converted to input splits and distributed to different nodes on the cluster. For this, Hadoop uses fair scheduling. Fair scheduling is a method of assigning resources to jobs such that all jobs, on average, get an equal share of resources over time. When there is a single job running, the job uses the entire cluster. When multiple jobs are submitted, each job gets roughly the same amount of time to convert a part of its job to input splits and distribute it among the nodes. As a consequence of fair scheduling, short jobs finish in a reasonable amount of time while not starving for long jobs to finish the HDFS phase.

- MapReduce: After the job completes the HDFS phase, it enters the MapReduce phase. Here, FIFO scheduling is followed and the scheduler is non preemptive. The job that completes HDFS first enters the MapReduce phase first. The next job that follows to the MapReduce phase must wait until the current job finishes its execution, following the principle of FIFO.

3.3 Simple Word Count Application

In this study, we use Hadoop word count as an example application to be running on the edge cloud site. This application counts the frequency of words present in a file and saves the count along with the words in an output file. The client sends an input file to the edge cloud, which processes the file and sends back the output result file back to the client. Basically, there are 3 steps in this process when the client requests to compute the frequency of words in a file:

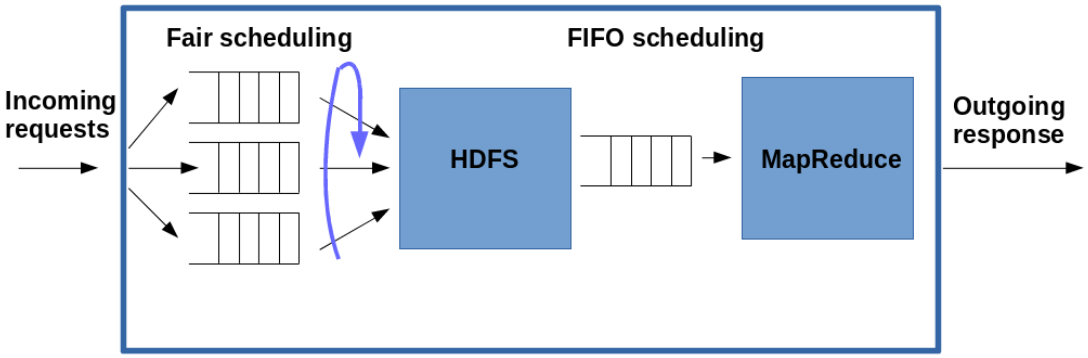

Figure 3.1: Hadoop job scheduler.

- File is transferred over a link to a particular edge cloud site, running Hadoop.
- At the cloud site:
    1. HDFS - The input file is converted into splits or chunks and distributed among the nodes of the cluster.
    2. MapReduce - MapReduce program is run to count the occurrence of words for each split at each mapper, (key,value) pairs are produced. These results are aggregated from all the mappers and sorted. A final count of words from all the mappers is made in the reducer which stores it in an output file. This output file is placed back on HDFS.
- Output file that is generated is sent over the link to the client.

The time taken to complete these three steps contributes to the response time for any given client request.

3.4 Modeling the Hadoop Cluster

In this study, the Hadoop cluster is modeled as an M/M/1 queuing system, considering the entire cluster to be a single unit. In this type of a system, the inter arrival times are exponentially distributed (Poisson arrivals), with average inter arrival rate λ. The service time is the time taken to completely process the job in the cluster and it has an exponential distribution, with average service rate μ. The service time is exclusive

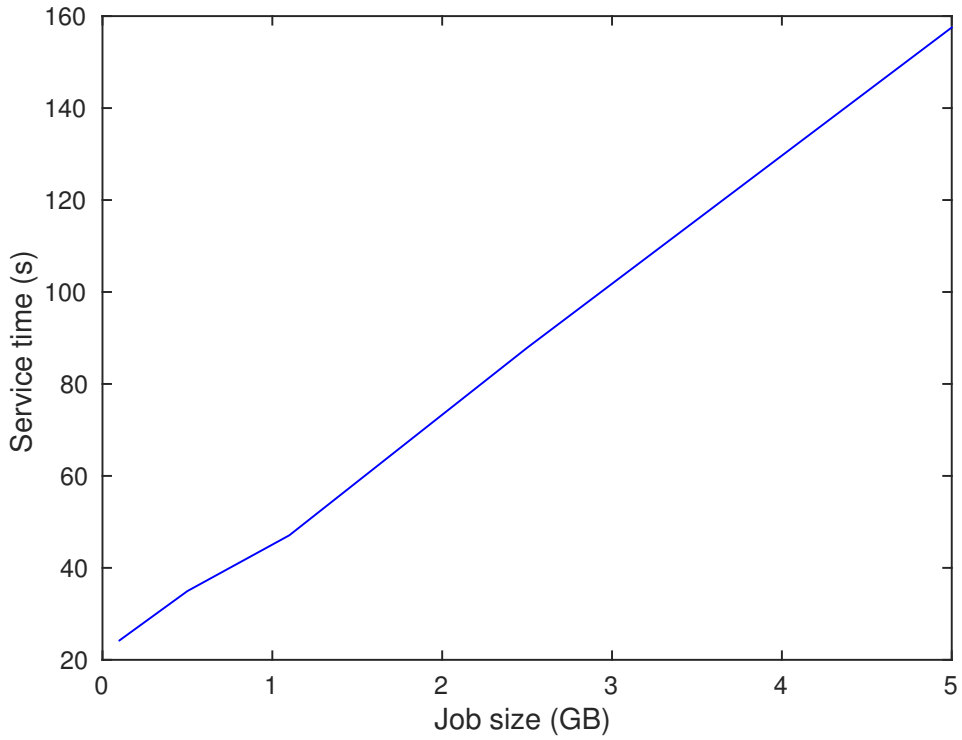

Figure 3.2: Service time distributions for various job sizes.

of the waiting time in the cluster. Figure 3.2 shows the service time for different job sizes and it is seen to be varying linearly with increase in job size. This means that if the job sizes were exponentially distributed, service times would also be exponentially distributed, making it an M/M/1 system. Waiting time is the sum of time spent by the job in queue prior to the HDFS phase and the time spent in queue while waiting for its turn to enter into the MapReduce phase after completing the HDFS phase. The buffer is assumed to be infinite and the queuing discipline is first in first out (FIFO).

Cluster utilization factor ρ is defined as the ratio of average inter arrival rate by the average service rate:

$$\rho = \lambda/\mu. \tag{3.1}$$

It represents the average proportion of time in which the cluster is occupied.

We measure the average job completion time at the edge cloud site which is the total time taken to process a job and this is inclusive of the waiting times in the queues. Job completion time is defined as the total amount of time a job spends in the system.

$$\text{Job completion time} = \text{Waiting time} + \text{Service time}. \tag{3.2}$$

This average job completion time reflects the characteristics of the cluster such as number of nodes, workload and processing capacity [16], rightly capturing the performance of the edge cloud and is hence, used as one of the parameters for ASR.

3.5 Synthetic Traffic Generator

In this work, a synthetic traffic generator is designed to measure the performance of a Hadoop cluster. It is used to induce traffic at the edge cloud and to study the effect of workload or traffic on the job completion time for different jobs. It is seen that as the traffic increases, waiting time in the system increases as MapReduce uses FIFO scheduling (waiting time for a new customer is the sum of remaining service time for jobs that are already in the queue) and thus, the job completion time which is a sum of service time and waiting time, also increases with increase in traffic.

Synthetic traffic generator is used in different experimental setups to induce traffic at the edge cloud site and to study the effect of inter arrival times on average job completion time. This is further discussed in detail in section 4.4.

Chapter 4

Application Specific Routing

There is a need to add application awareness to improve the user's QoE and to do so, we need to make routing decisions based on parameters that are application specific. For instance, a user watching videos may need more bandwidth than a user playing games but the latter may need lower latency. Routing decisions that are taken on the basis of such application specific parameters can not only help in improving user's QoE but can also aid in resource management.

In this work we propose a new method of routing called the Application Specific Routing on top of MF VN which can use different parameters, such as, energy used at cloud sites, cost of spawning new servers at cloud sites, high bandwidth requirements, low latency requirements, time of the day, workload at the cloud sites and the like, to make routing decisions.

For real time applications such as for CPS or IoT, typically a large amount of data is sent from the user to the cloud site where processing (usually, heavy computation) takes place. The processed data is then sent back to the user. For applications like these, link performance and edge cloud performance greatly affect user's QoE because link performance affects the transfer of the large amount of data, while, edge cloud performance affects the processing of big data. In the following section, we study the parameters that affect link and edge cloud performance, and figure 4.1 tabulates the details of the relevant software modules used in this study.

4.1 Link Performance Metric

Link performance metric is a measure of network performance which is affected by various parameters, like:

Software module	Version	Operating system	Type	Purpose	Applicati-on written in	Others
Netcat	1.10	Ubuntu	Network utility	File transfers	N/A	Protocol – TCP/IP, Cubic version
netEm	3.13.0-53	Ubuntu	Network emulator	Vary link performance	N/A	
Hadoop	2.4.1	Ubuntu	Distributed file system	Parallel and distributed computing on edge cloud	Java	Modules-HDFS & MapReduce. Fair scheduler used
Synthetic traffic generator	N/A	Ubuntu	Customized application	Induce workload at edge cloud	Python	Inter arrival times of requests: exponentially distributed
MF sender and receiver	N/A	Ubuntu	Customized application	File transfers on MF	C++	

Figure 4.1: Details of the relevant software modules used in this study.

- Bandwidth - Maximum rate at which data can be transferred.
- Latency - Delay with which a packet is transmitted from one point to another. Several factors contribute to latency such as the processing at routers and transmission delay(time it takes for a packet to be transmitted at speed of light).
- Jitter - Variation in the time of arrival of the packets at the receiver's end.
- Bit Error Rate (BER) - Number of bits of a data stream that have been received in error due to noise, interference, distortion or bit synchronization divided by the total number of received bits.

In this work, link performance incorporates both bandwidth and latency as they significantly affect the network performance and we study their effect on the network link. Files are transferred from a client to a single hop cloud site as shown in figure 4.2 (the block diagram for the same is shown in figure 4.3)for different job sizes: 1 MB, 10MB, 100MB, 500 MB, 1 GB, 2.5 GB and 5 GB to study transfer time under

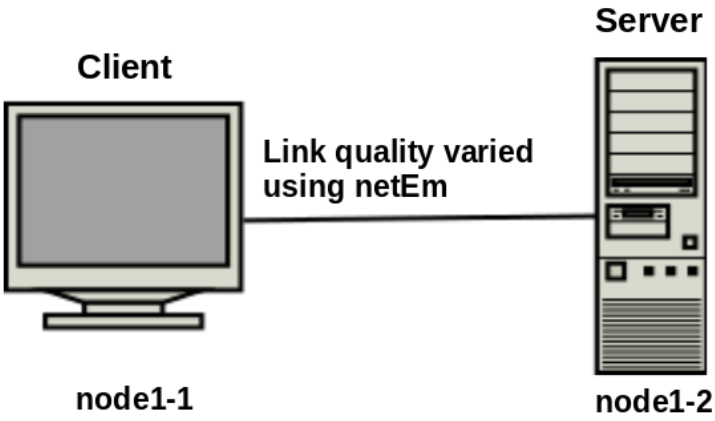

Figure 4.2: Experimental setup to study the effect of latency on transfer time.

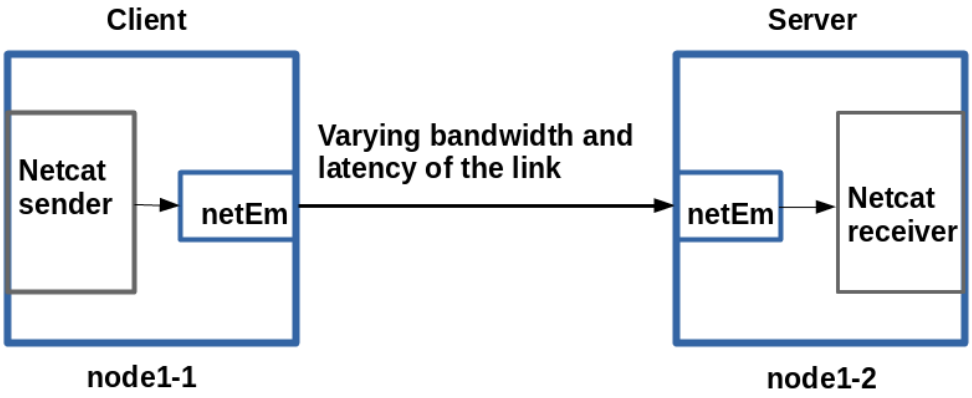

Figure 4.3: Block diagram for the experimental setup in figure 4.2.

different latencies(RTT): 15 ms, 30 ms and 60 ms and bandwidth: 1000 Mbps, 500 Mbps and 100 Mbps. These input parameters are chosen in accordance with real time network scenarios wherein a client connects to an edge cloud site through a wireless access point. A network emulator tool called 'netEm' is used to vary the link latency for this experimental setup [17]. Buffer size of 16 MB and TCP CUBIC version is used along with TCP auto scaling feature [18]. CUBIC is suitable for high speed network environments because the window growth rate is independent of RTT which keeps TCP friendly under both short and long RTT paths [19].

The experimental objectives were to study the effect of latency on transfer time for different job sizes. The experiment was repeated for varying link bandwidths. Files were transferred across the link for different network latencies and transfer times were measured. Figures 4.4 and 4.5 show the effect of latency on transfer time on a Gigabit

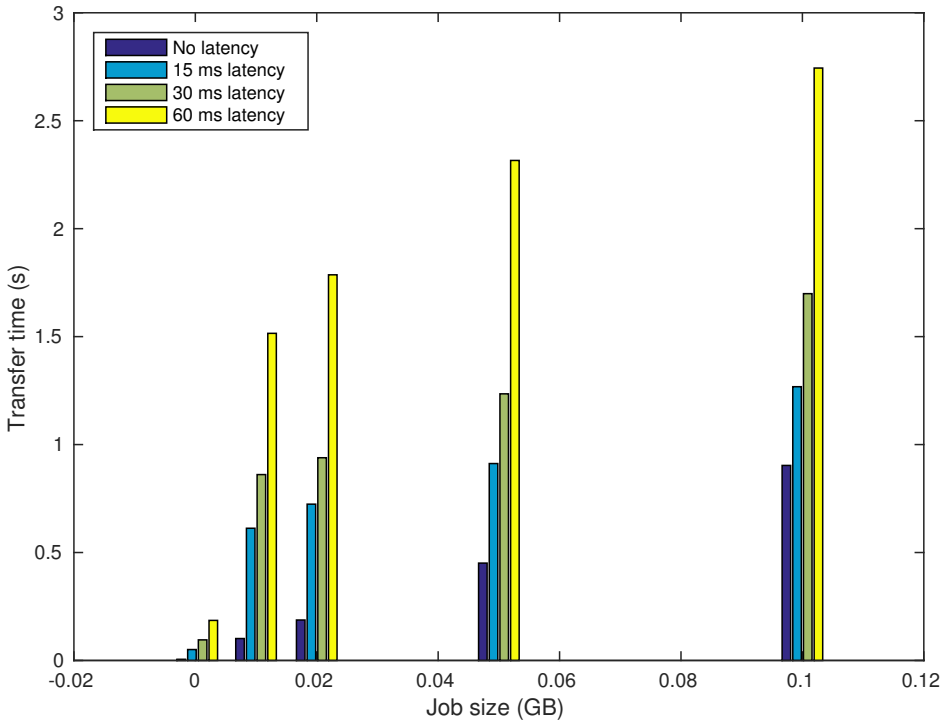

Figure 4.4: Effect of latency on transfer time for small data size on a link bandwidth of 1000 Mbps.

link for small and large data sizes respectively. Similarly, figures 4.6 and 4.7 show the effect of latency on transfer time on a 500 Mbps link for small and large data sizes respectively . It is seen that, with increase in data size, effect of latency on transfer time also increases. There are 3 observations to be made:

- When the data size is small, transfer time is affected by the slow start of TCP/IP protocol, thus, with increase in data size, transfer time also increases significantly. We see this effect for job size of up to 100 MB as shown in figures 4.4 and 4.6.

- As the data size increases transfer time is almost about the same for different latencies because large amount of data is transmitted through the link to mask latency affects, shown in figures 4.5 and 4.7. This is performed by TCP/IP auto scaling. The amount of data sent out by the sender depends on factors like congestion window size and receiver buffer size. When receiver buffer size is large, receiver window size advertised is also large and thus, large amount of data is sent

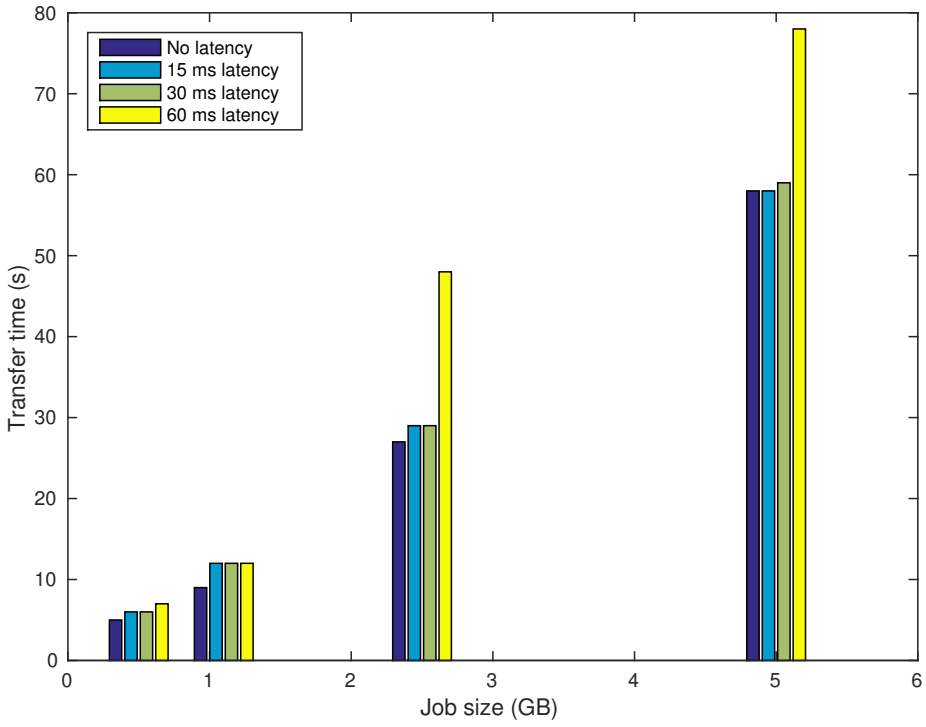

Figure 4.5: Effect of latency on transfer time for large data size on a link bandwidth of 1000 Mbps.

across the link to mask latency effect.

- With very large job sizes and large latencies, transfer time is severely affected. This is because the amount of data to be sent from sender is either affected by capacity of link or because of the application's slow retrieval rate from the buffer on the receiver end. If the application's retrieval rate is slow, then the advertised receive window size is small, so, the sender cannot send enough data to mask the latency effects because of which transfer time is affected significantly [20]. This is explained in detail in section 4.2.2.

We also study the effect of bandwidth on transfer time for different job sizes with the same setup as in figure 4.2 and the results are in figure 4.8. The transfer time depends on both bandwidth and job size. It is seen that, for any given job size, time taken to transfer a file on a 100 Mbps link is almost ten times and for a 500 Mbps link, it is twice of that on a 1000 Mbps link. This is because bandwidth determines the

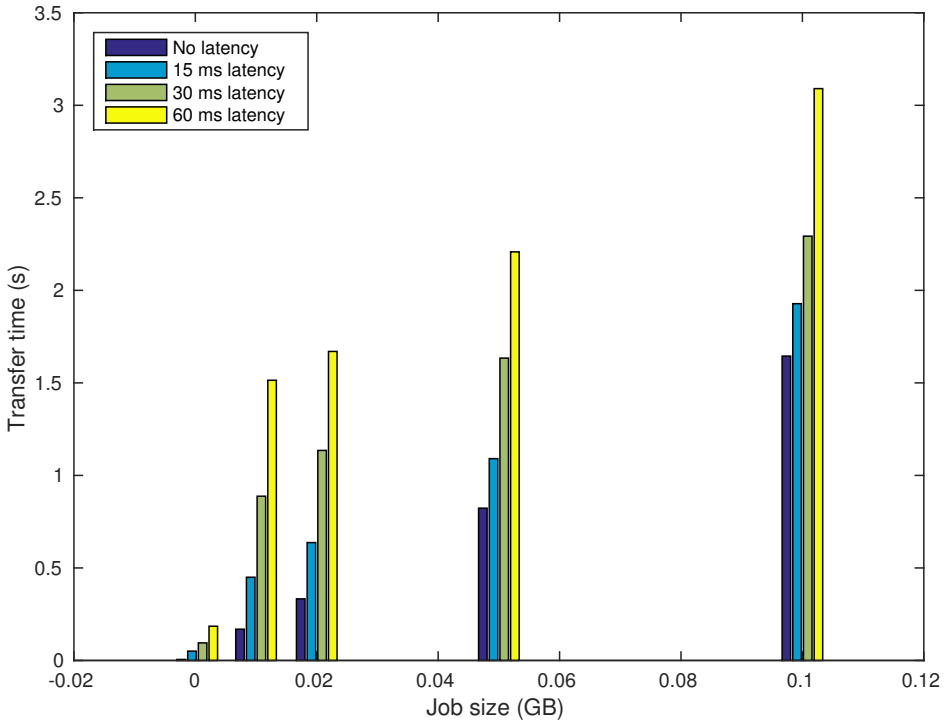

Figure 4.6: Effect of latency on transfer time for small data size on a link bandwidth of 500 Mbps.

rate at which data can be transferred on a link. So, higher the bandwidth, higher the data transfer rate and hence, lower the transfer time and vice versa. Bandwidth is an essential parameter to be considered especially when considering big data for real time applications as it plays a major role in determining the transfer time. Thus, bandwidth is an important parameter to be considered for link performance.

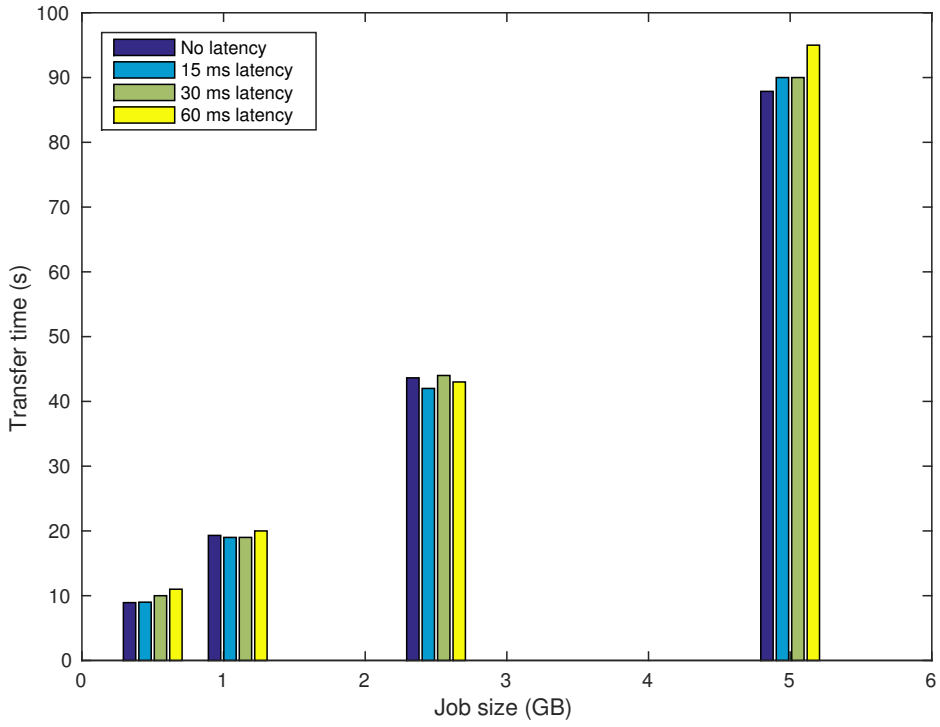

Figure 4.7: Effect of latency on transfer time for large data size on a link bandwidth of 500 Mbps.

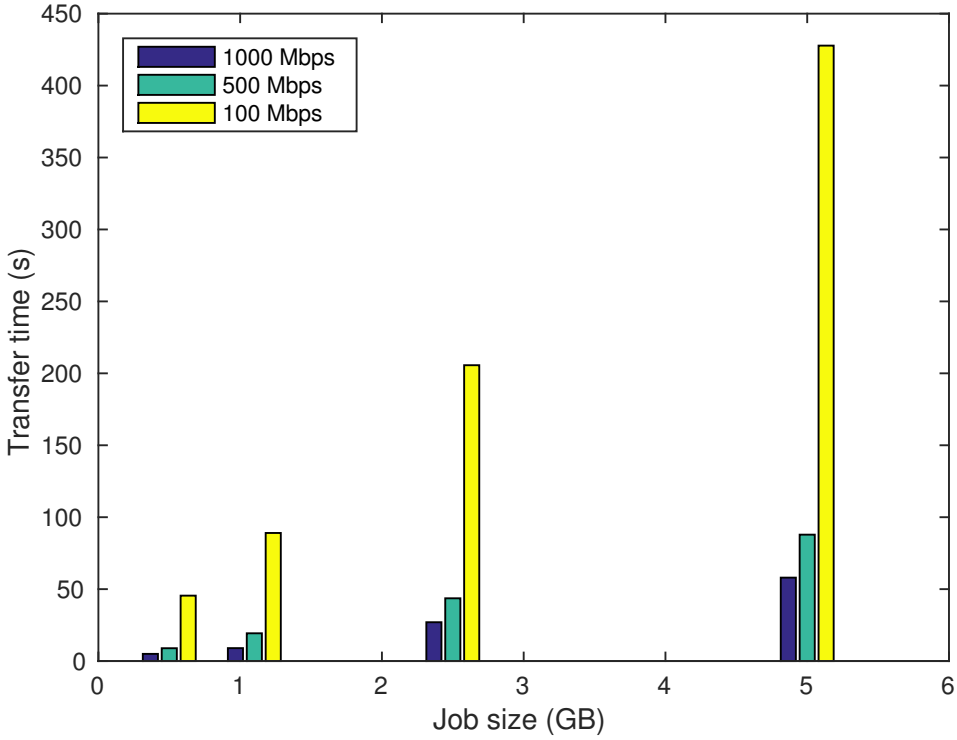

Figure 4.8: Effect of bandwidth on transfer time for different job sizes.

4.2 Effect of Link Performance on Response Time

We study the effect of link performance on response time by conducting a set of experiments, wherein the link quality between client and the edge cloud is changed with respect to bandwidth: 1000 Mbps, 500 Mbps and 100 Mbps and latency: 15 ms, 30 ms and 60 ms. Edge cloud with a cluster of 9 nodes, runs Hadoop software for parallel and distributed computing. The response time for a client's request is given as a sum of input file transfer time, job completion time at the edge cloud site and output file transfer time as shown in figure 4.11. Simple word count application runs on the edge cloud and a buffer size of 16 MB is used for the following set of experiments. The experimental setup is shown in figure 4.9 and the block diagram for the same is shown in figure 4.10.

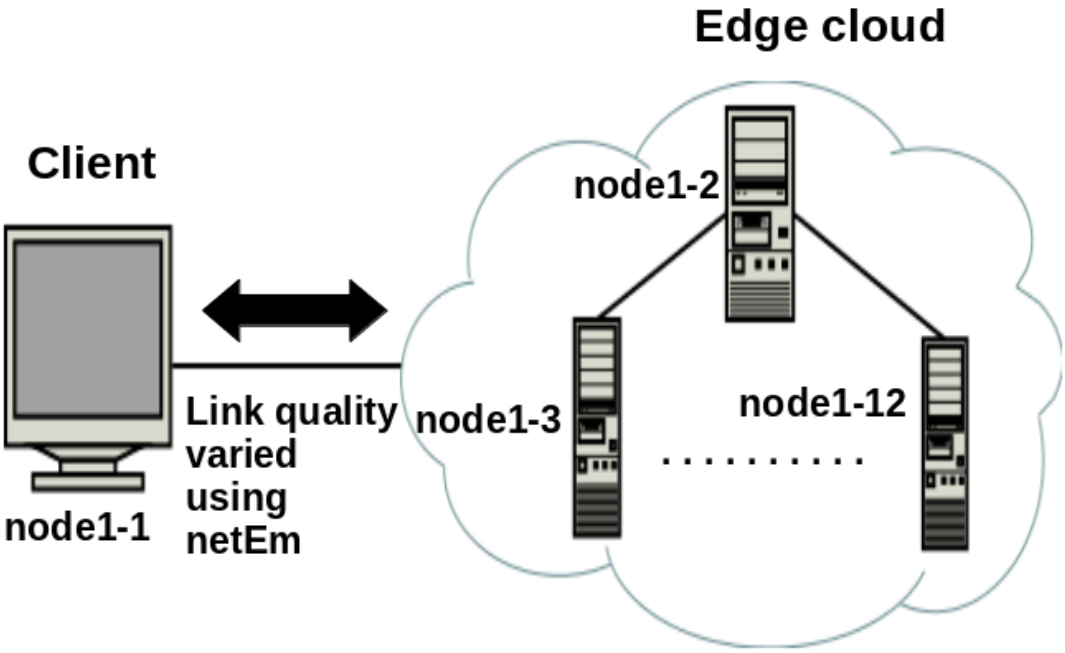

Figure 4.9: Experimental setup to study the effect of link performance on response time.

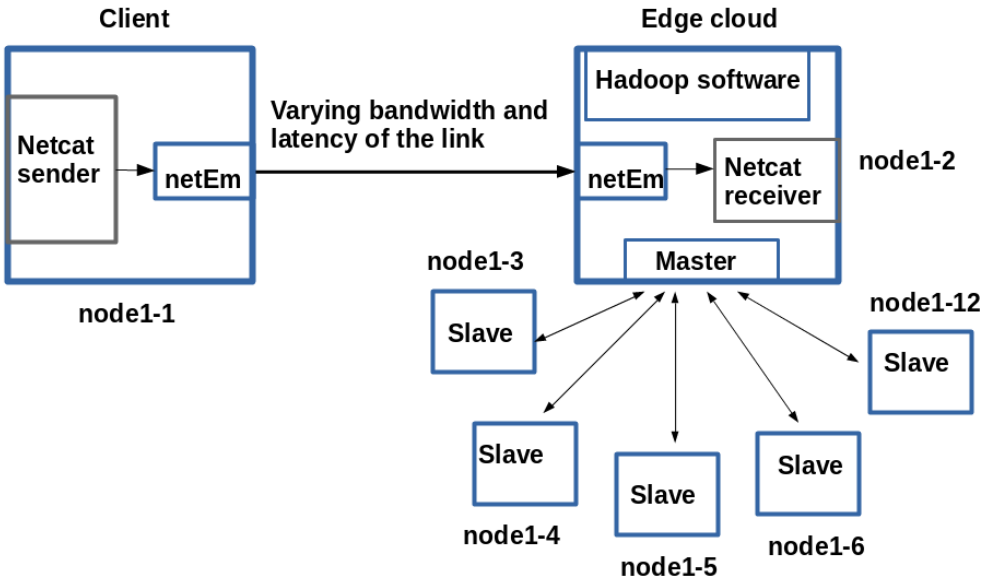

Figure 4.10: Block diagram for the experimental setup in figure 4.9.

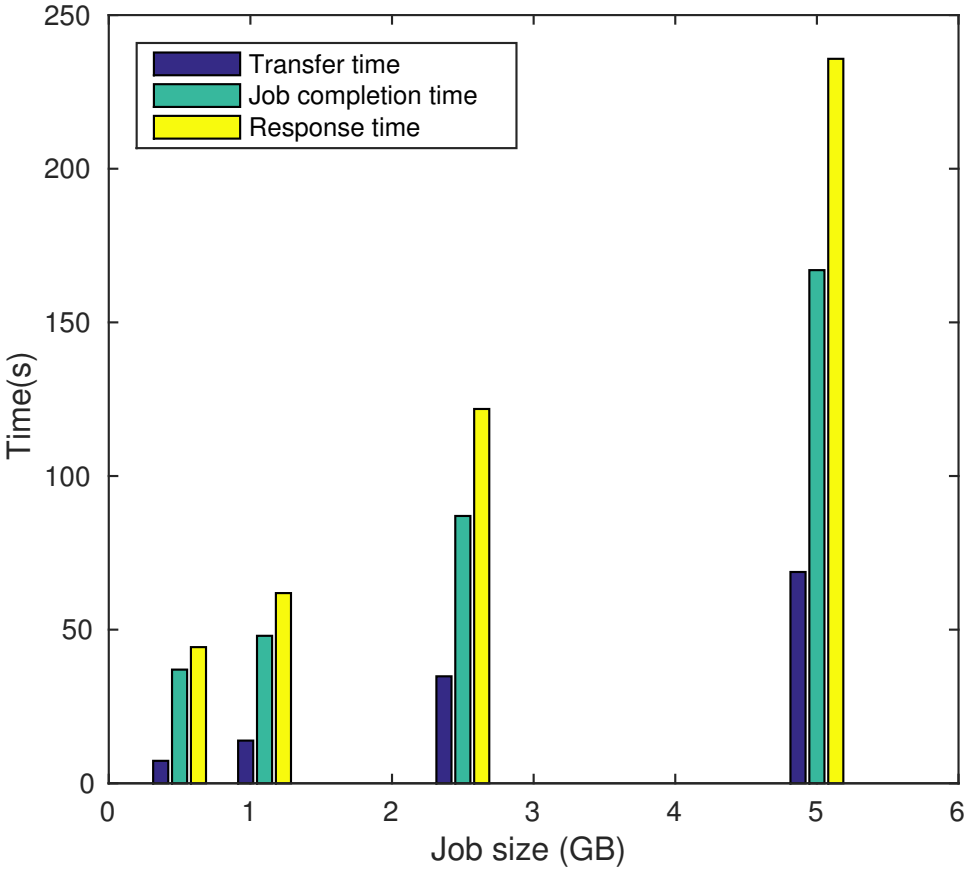

Figure 4.11: Study of response time for different job sizes on a Gigabit link.

4.2.1 Variation in bandwidth of links

Figure 4.11 shows the effect of transfer time and job completion time on the response time, for different job sizes on a 500 Mbps link. The job completion time at the cluster is a sum of HDFS and MapReduce time and they vary according to the job size. Figure 4.12 shows the effect of bandwidth on the response time for a job of size 2.5 GB. Variation in the bandwidth of the link between the client and the edge cloud affects the file transfer times which contributes to the total response time for a user's request. However, the HDFS and MapReduce time at the cluster remain the same for different bandwidth links because it is only affected by the job size.

It is seen that time taken to transfer a file on a 100 Mbps link is almost ten times of that on a 1000 Mbps link for the same job size and transfer time contributes to the

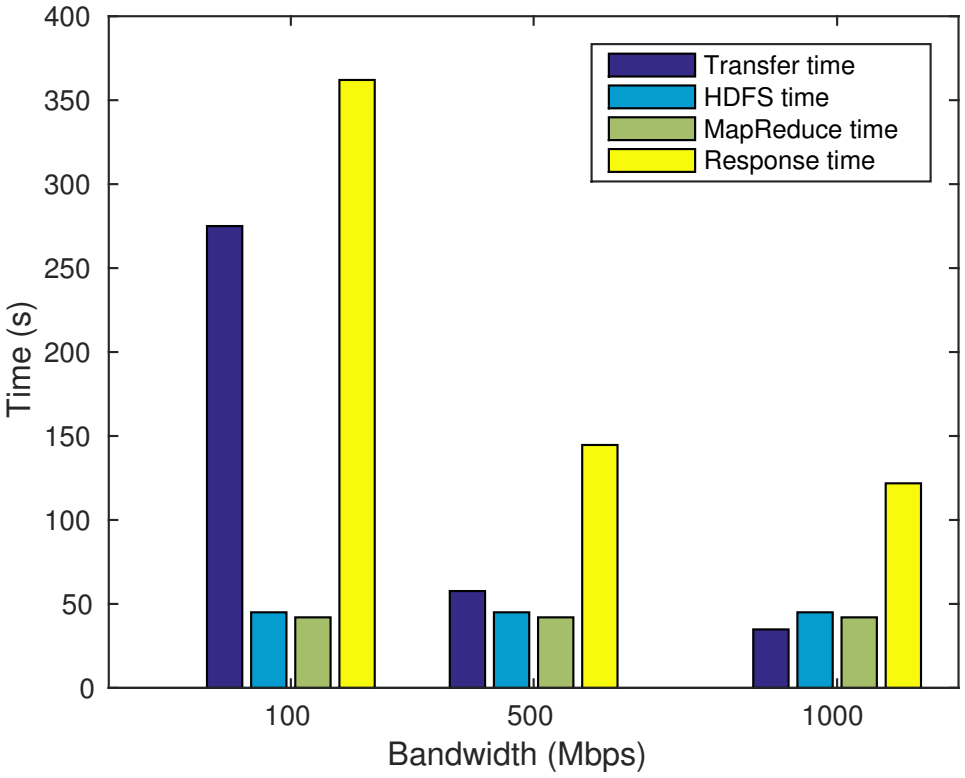

Figure 4.12: Effect of bandwidth on response time for a job size of 2.5 GB

response time. Thus, bandwidth is an important parameter to be considered for link performance which will contribute to the ASR metric.

4.2.2 Variation in latency of links

The setup to study the effect of latency on response time is the same as mentioned in 4.2.1. Figure 4.13 shows the effect of latency on response time for a job of size 2.5 GB. It is seen that the file transfer time increases as the latency of the link increases.

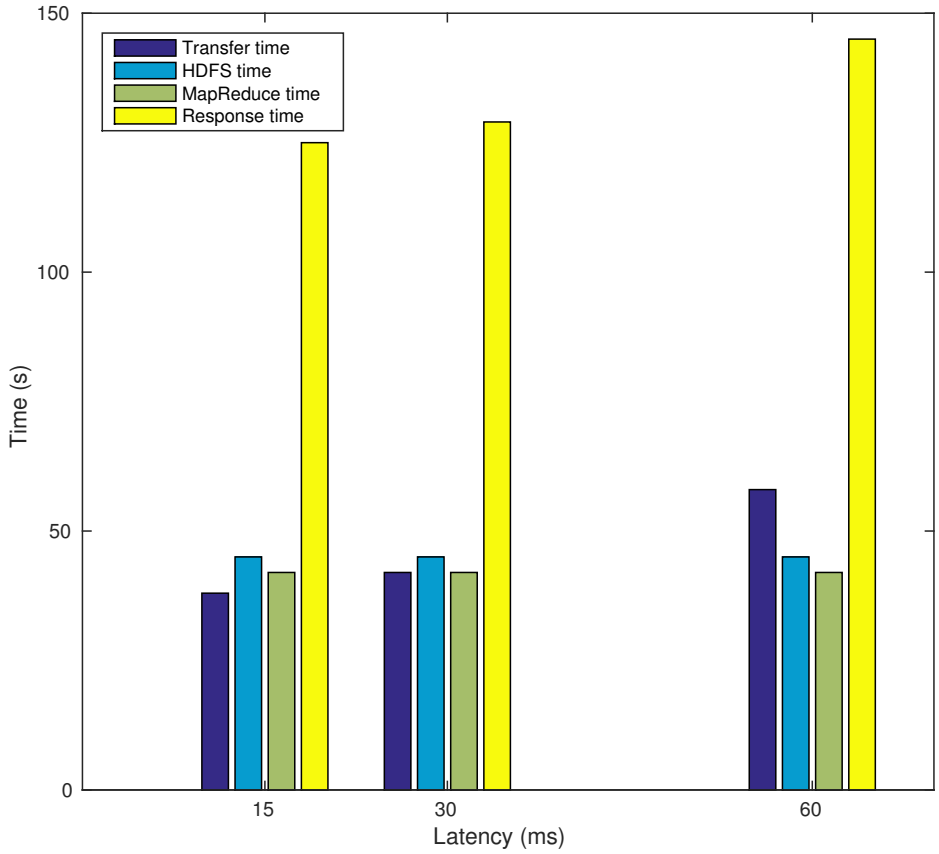

Figure 4.13: Effect of latency on response time for a job size of 2.5 GB on a Gigabit link.

To optimize TCP throughput for scenarios where the bandwidth is high or there is high latency between client and edge cloud (called as long fat pipes), the client should send enough packets to fill the fat pipe between the sender and receiver. The capacity of the logical pipe can be calculated by:

Capacity in bits = Path bandwidth in bits per second * Round-trip time (seconds). (4.1)

Links with high bandwidth and large latency have high BDP. High BDP links need large window sizes and for these window sizes, RFC 1323 [21] defines window scaling

that allows receivers to advertise a window size larger than 65,535 bytes. A TCP window scale option includes a window scaling factor that, which when combined with the 16-bit window field in the TCP header, can increase the receive window size to a maximum of approximately 1 GB. Both TCP peers can indicate different window scaling factors to use for their receive window sizes. By allowing a sender to send more data on a connection, TCP window scaling allows TCP nodes to better utilize some types of links with high BDPs and provides higher throughput.

From figures 4.5 and 4.7, it is seen that the transfer time is strongly affected by latency for 1000 Mbps, 60 ms latency link because the buffer size is 16 MB and the bandwidth delay product (BDP) is about 7.5 MB , whereas for 500 Mbps, 60 ms link (BDP is 3.75 MB) and 500 Mbps, 30 ms link (BDP is 1.875 MB), transfer time is about the same because the buffer size is much greater than the amount of data to be transferred every RTT. TCP stack doesn't allow data to be sent if there is no room in the receiver's buffer for the incoming data, therefore, as latency increases, amount of data to be sent on the link also increases to compensate for the latency, but, if there is no room in the receiver's buffer then the amount of data to be sent will reduce depending on the receiver buffer. The receiver's buffer becomes the bottleneck in determining the amount of data that must be in flight over the link to compensate for latency [20].

Although the receive window size is important for TCP throughput, another factor for determining the optimal TCP throughput is the application's retrieval rate from the receiver buffer. If the application does not retrieve the data quickly, the receiver buffer can begin to fill, causing the receiver to advertise a smaller receive window size. In the extreme case, the entire maximum receive buffer is filled, causing the receiver to advertise a window size of 0 bytes. In this case, the sender must stop sending data until the receive buffer has been cleared. Therefore, to optimize TCP throughput, the TCP receive buffer for a connection should be set to a value that reflects both the BDP of the connection's transmission path and the application's retrieval rate.

Latency affects transfer time and is an important parameter for link performance, hence, contributes to the ASR metric.

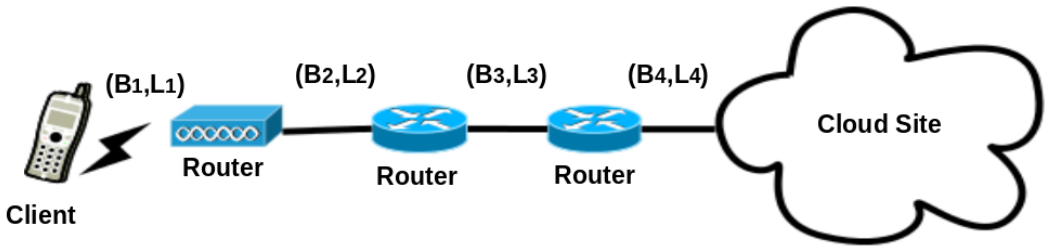

Figure 4.14: Experimental setup to study the effect of link performance on response time in MobilityFirst.

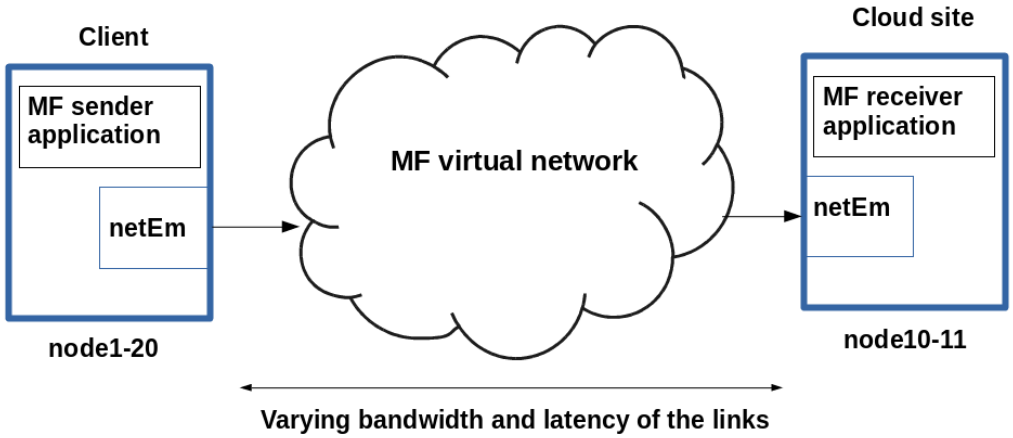

Figure 4.15: Block diagram for the experimental setup in figure 4.14.

4.3 Link Performance Metric on MobilityFirst

We also study how transfer time is affected by link quality on MF. Link quality is varied by adjusting bandwidth and latency (using netEm tool) which affects MF's SETT values. In these set of experiments, we transfer the same set of files as mentioned in section 4.2 on MF virtual network to study the effect of bandwidth and latency on transfer time.

The experimental setup is shown in figure 4.14, where (B, L) represents the *bandwidth* and *latency* respectively, of each link on the path from the client to the edge cloud. The block diagram for the same is shown in figure 4.15. The file to be transferred is divided in to messages of size 1 MB and the buffer size used here is 10 MB. Figure 4.16 shows the effect of bandwidth on transfer time, while figures 4.17 and 4.18 show

the effect of latency on transfer time. Like mentioned is section 4.1, bandwidth plays a crucial role in determining the transfer time as it influences the rate at which data must be transferred over the link. Hence, transfer time on a 100 Mbps link is at least four times as that on 500 Mbps link.

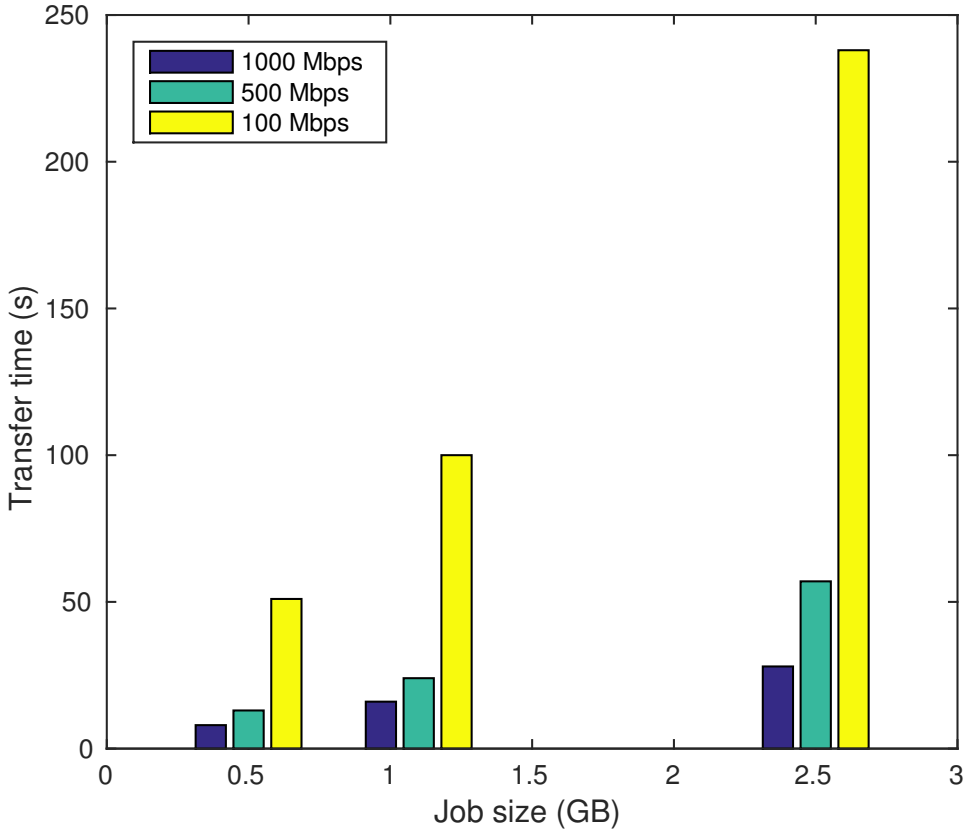

Figure 4.16: Effect of bandwidth on transfer time on a MobilityFirst Virtual Network.

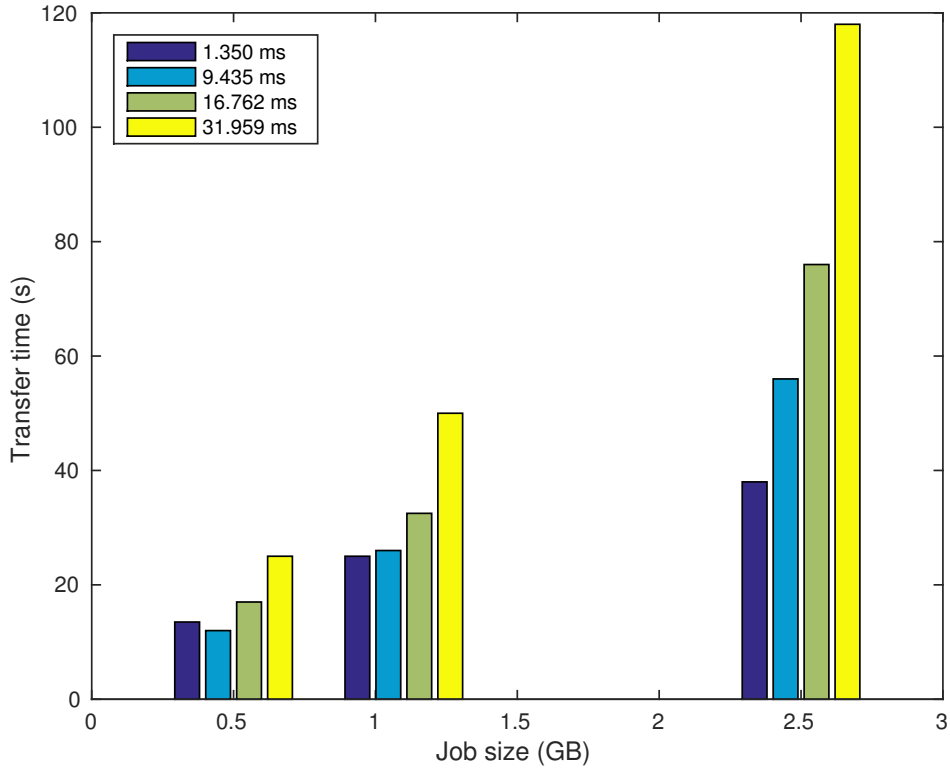

Figure 4.17: Effect of latency on transfer time on a MobilityFirst Virtual Network using Gigabit links.

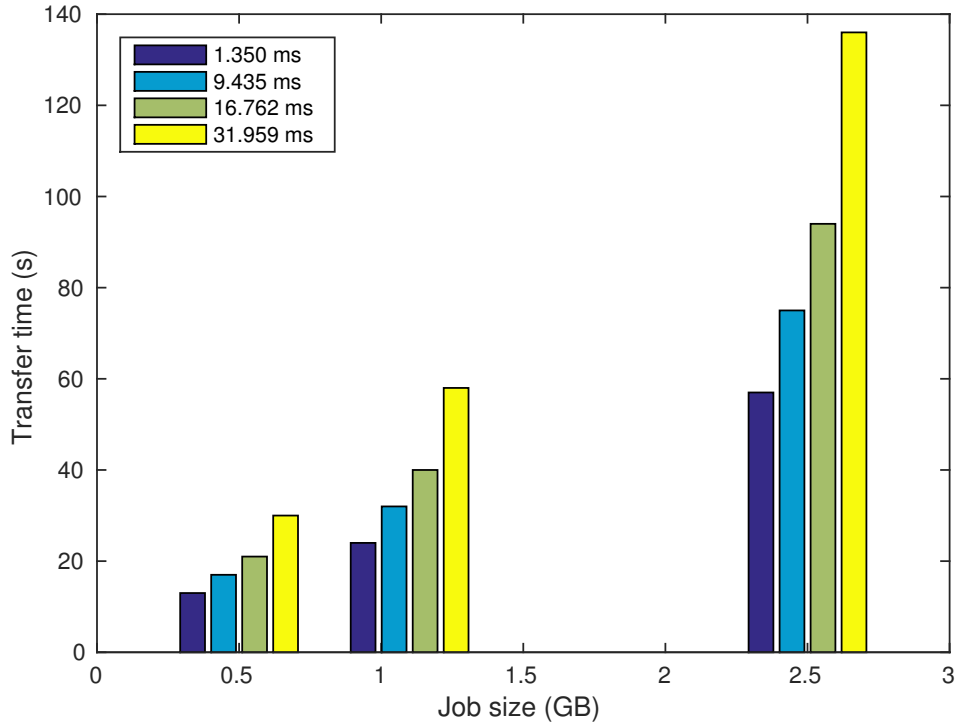

Figure 4.18: Effect of latency on transfer time on a MobilityFirst Virtual Network using 500 Mbps links.

Figures 4.17 and 4.18 show the effect of latency on transfer time on an MF network. The flow control mechanism on MF in still under development. When a chunk of data needs to be sent over the link, the next chunk of data is not sent until an acknowledgment is received for the previous chunk. Hence, latency affects transfer times more than expected in these set of experiments though this is likely to be reduced in future releases of the MF code with better optimized link layer protocol.

4.4 Edge Cloud Performance Metric

Edge cloud performance metric is another metric that is chosen to determine the best destination/ edge cloud to service a request and to ensure good QoE for a client. Each edge cloud site is characterized by its utilization (ρ). When ρ is high, it implies that the work done by the nodes in the cluster is high which is because of high traffic intensity, thus, ρ directly reflects the traffic intensity at the edge cloud. For instance, if ρ at

an edge cloud is high, say 0.8, then, the job completion time at this cluster would be much higher than at an edge cloud with $\rho = 0.4$. As the traffic intensity increases, the amount of time a new client request has to wait until it is serviced increases and thus the total time taken for the entire job to be processed by the cluster also increases. Job completion time can be defined as the time a client has to wait until the job is serviced; including both service time and waiting time in the cluster. Thus, the average job completion time at the cluster reflects the traffic intensity or ρ at that cloud site.

The job completion time is indirectly affected by other parameters of the cluster such as; capacity of the servers, number of nodes in the cluster and the like [16] . An experiment is conducted to study the effect of number of nodes on job completion time in the cluster. From figure 4.19, we see that as the number of nodes in the cluster increases, processing time or job completion time in the cluster decreases; i.e., job completion time and the number of nodes in the cluster are inversely proportional. As the number of nodes increase, there are more nodes available to process the job in parallel because of which the overall job completion time decreases. This is seen in figure 4.19. However, if the number of nodes in the cluster is more than required for a given job size, communication overhead among the nodes dominate over the processing time and subsequently, the job completion time in the cluster increases after a certain point.

In another experiment, we study the effect of processing speed of the servers on the job completion time. The ORBIT [22] testbed's GRID and Sandbox 9 (SB9) have been used for these set of experiments. GRID nodes have CPU speed varying from 970 MHz to 3400 MHz, while SB9 have CPU speed of 2300 MHz. SB9 provides lower job completion time since it has better performance in terms of average CPU speed as compared to the GRID nodes and this is seen in figure 4.20.

4.4.1 Effect of traffic intensity on average job completion time in an M/M/1 system

In this work, we study the effect of traffic intensity on average job completion for different inter arrival rates of the job requests.

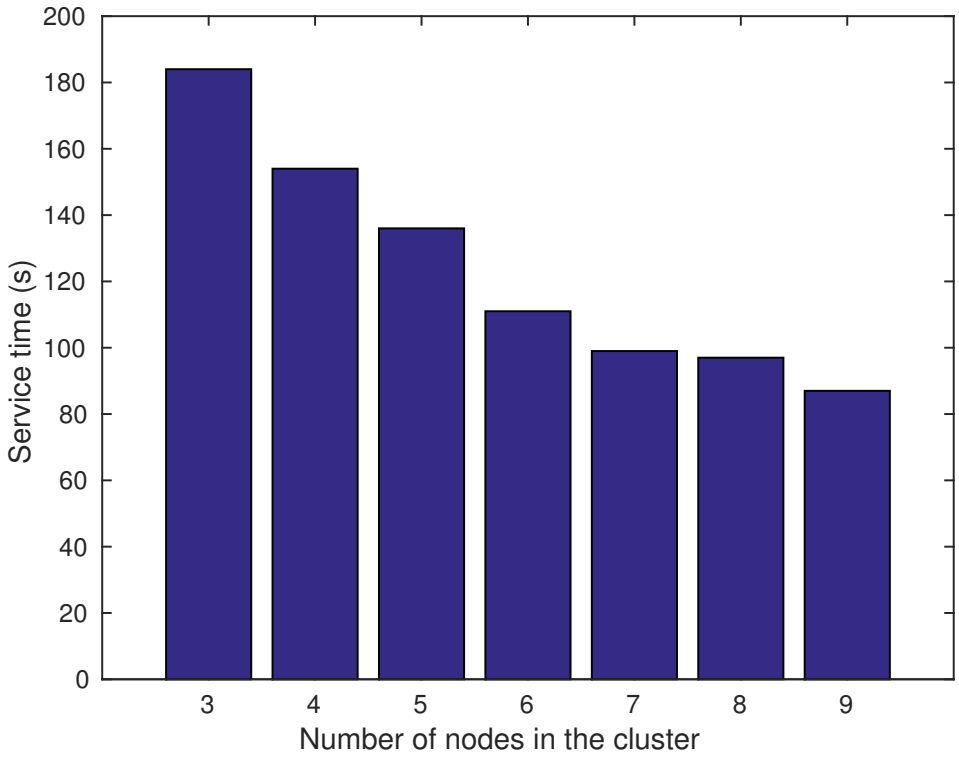

Figure 4.19: Effect of number of nodes on service time in a Hadoop cluster for a data size of 2.5 GB.

The average job completion time is given by:

$$\text{Average job completion time} = \frac{1}{\mu - \lambda}(seconds). \tag{4.2}$$

By calculating average service rate, $\mu = 0.01766$ (from figure 3.2), we study the effect of traffic or inter arrival rate of requests on average job completion time in the cluster, shown in figure 4.21. For an M/M/1 system the input job sizes and the probabilities of their requests need to be exponentially distributed in order to calculate their average service rate but here, we assume that the requests have equal probability and then calculate the average service rate because these parameters are highly application dependent.

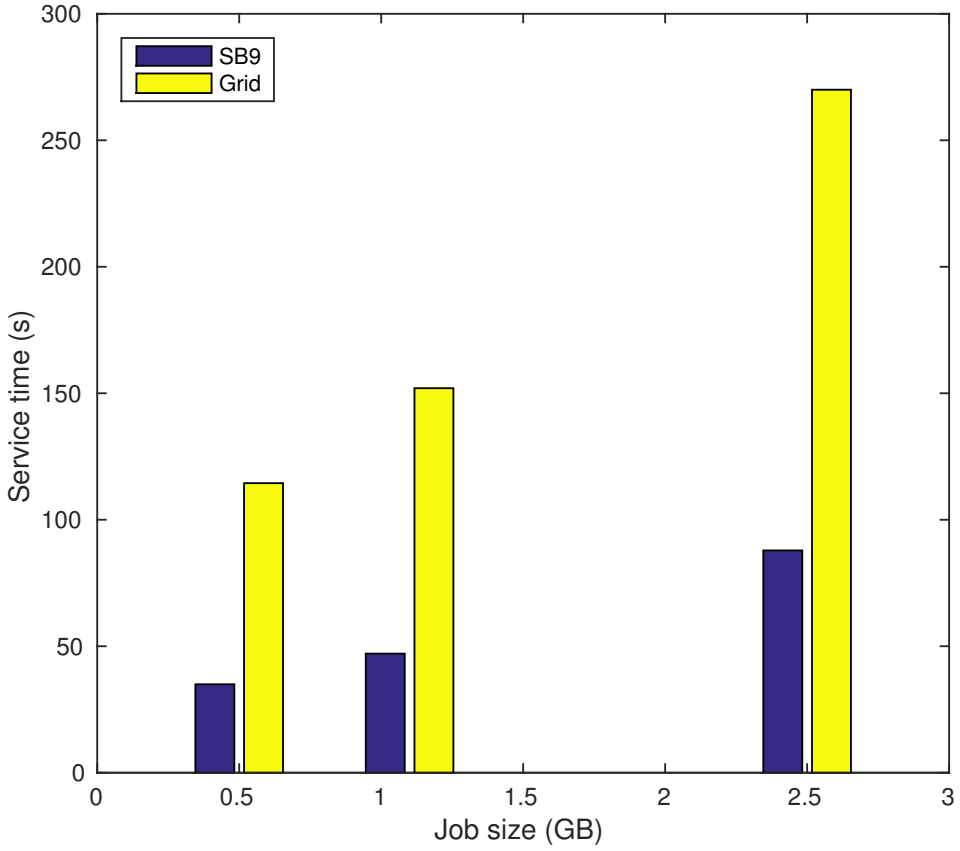

Figure 4.20: Effect of processing speed of different servers in a Hadoop cluster on service time.

4.4.2 Effect of traffic intensity on average job completion time in a Hadoop cluster

We study the effect of λ on average job completion time by setting up a Hadoop cluster of 9 nodes on ORBIT testbed. Traffic is induced at this edge cloud site by a synthetic traffic generator and we measure the job completion time for every client's request. The job completion time is averaged over a large number of requests and the effect of inter arrival rate on average job completion time is studied. Here, job sizes of 545 MB, 1.1 GB and 2.6 GB have been used. Figure 4.22 shows the effect of inter arrival rate on average job completion time and this performance study is done from zero initial state of the Hadoop cluster.

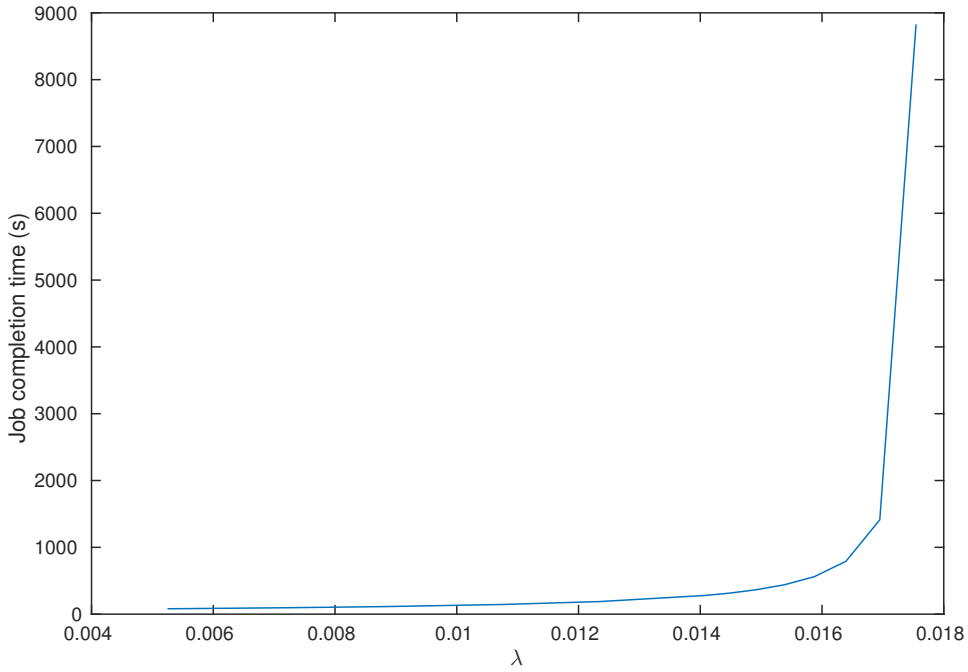

Figure 4.21: Effect of λ on average job completion time in an M/M/1 system.

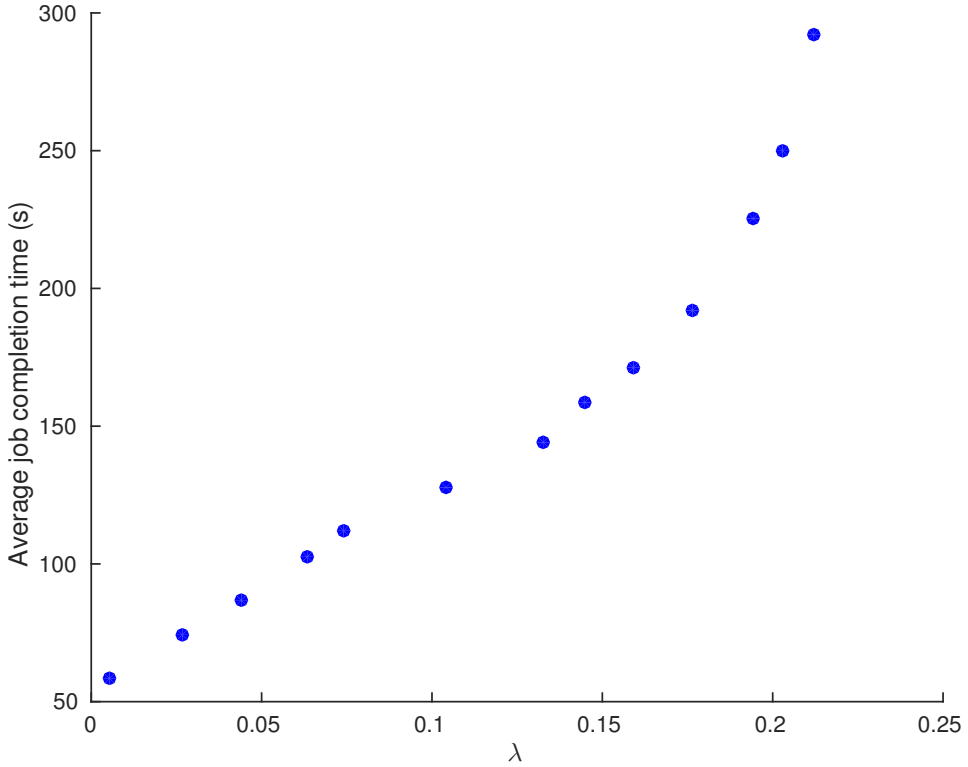

Figure 4.22: Effect of λ on average job completion time in a Hadoop cluster.

Chapter 5

System Level Evaluation

This work evaluates the combined effect of both access network and edge cloud performance on the response time, which is given by :

$$\text{Response Time} = \text{Transfer time} + \text{Job completion time at the edge cloud.} \quad (5.1)$$

5.1 Combined Effect of Performance of Link and Performance of an M/M/1 System on Response Time

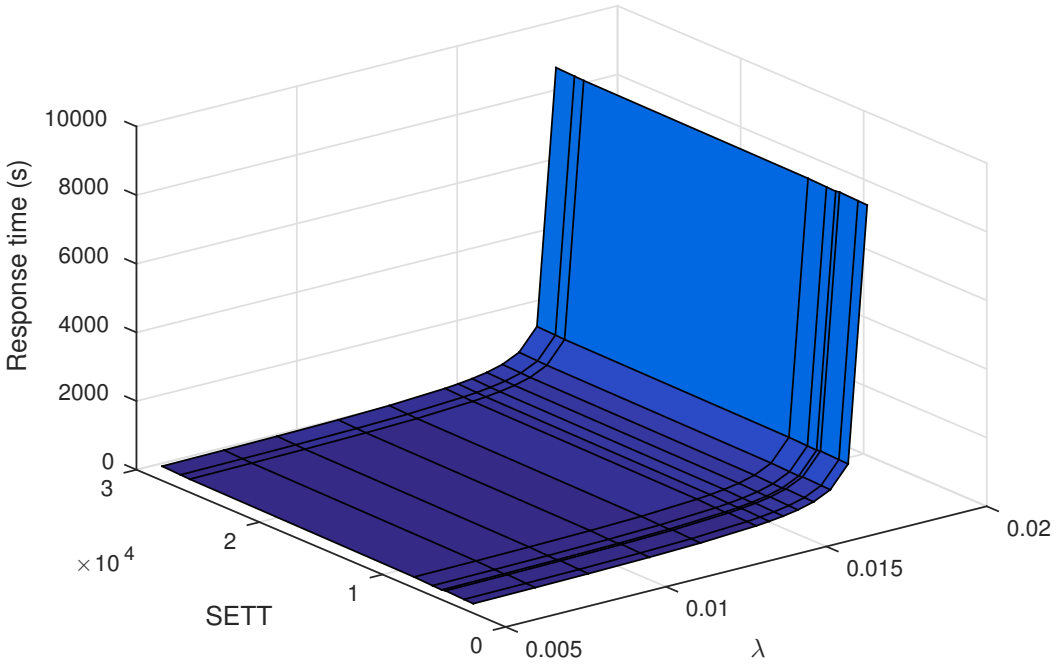

Figure 5.1: Effect of SETT and inter arrival rate (λ) on the response time in an M/M/1 system.

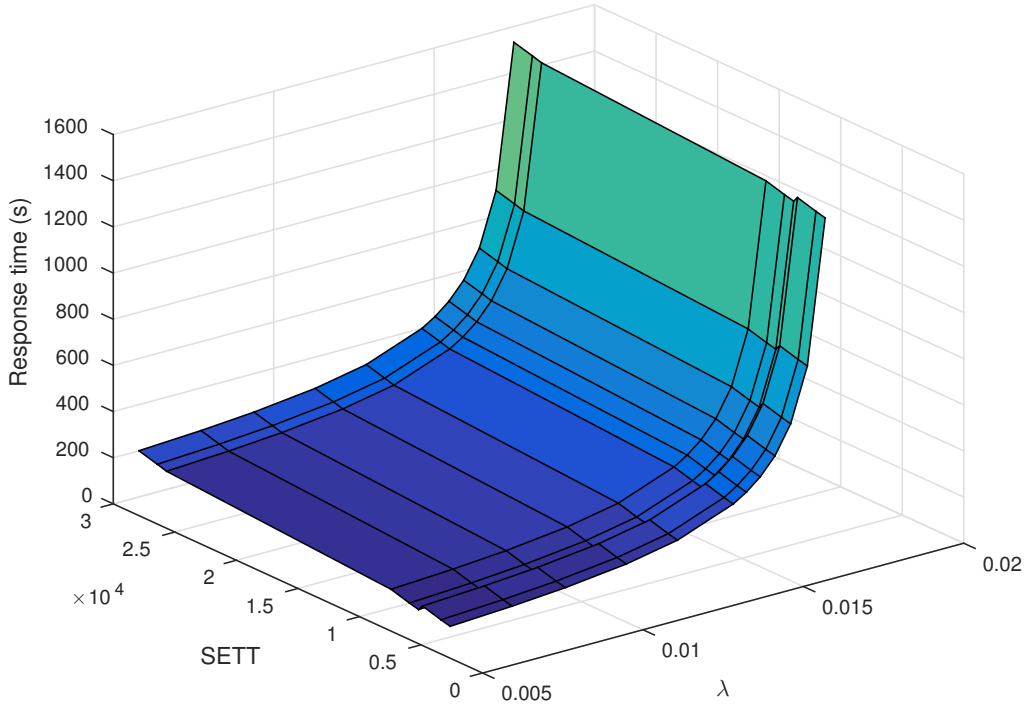

Figure 5.2: Effect of SETT and inter arrival rate (λ) on the response time in an M/M/1 system for $\lambda \leq 0.0169$

From figure 5.1, it is seen that response time steeply increases when $\lambda > 0.0169$, this is because the average job completion time of the system becomes very large as the inter arrival rate, λ approaches the service rate, μ. In order to provide the user with good QoE, this region of the graph needs to be avoided and we can do so by restricting the ASR algorithm to avoid destinations where λ is greater than 0.0169. Figure 5.2 shows the same plot with restricted λ values and this region looks smooth.

5.2 Combined Effect of Performance of Link and Performance of a Hadoop cluster on Response Time

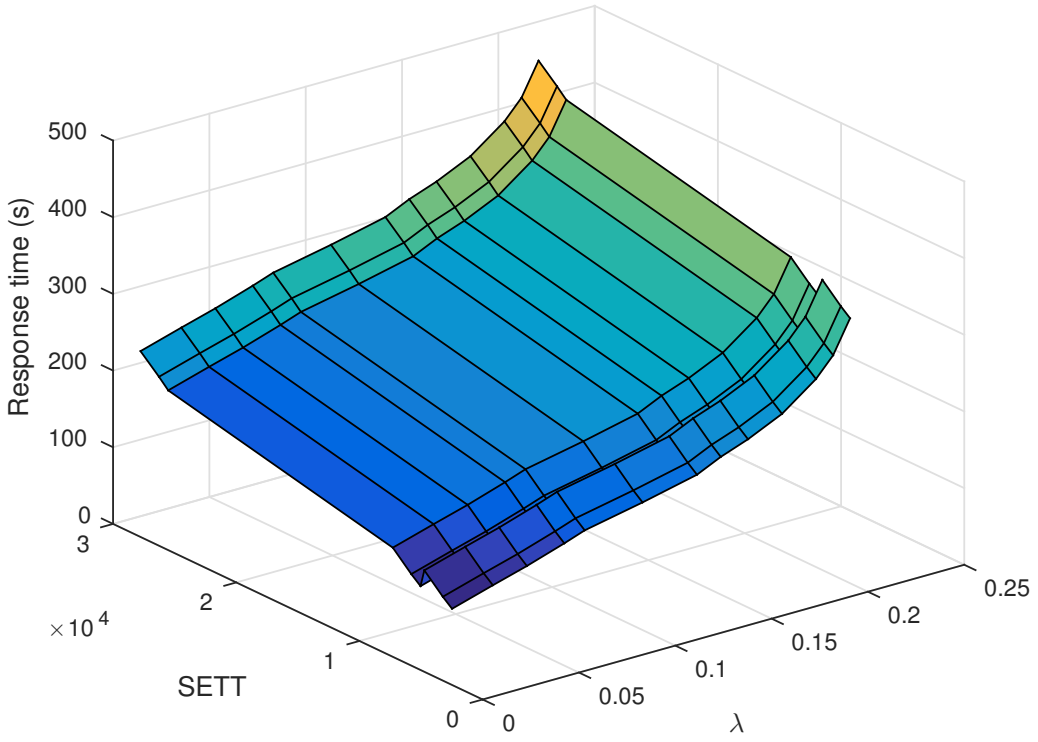

Figure 5.3: Effect of SETT and inter arrival rate (λ) on the response time in a Hadoop cluster.

From figure 5.3 it is seen that the region is smooth and there is no region which we particularly need to avoid to provide the user with the best QoE. Hence, for both figures 5.2 and 5.3, the decision metric can be written as :

$$\text{Estimated response time} = \text{Estimated transfer time} + \text{Average job completion time.} \tag{5.2}$$

The estimated transfer time is a function of client's file size or job size, while the average job completion time is dependent on the size of files or jobs that the edge cloud services. The file size plays a key role in determining the destination as it significantly

affects the link performance as well as the edge cloud performance and is considered as an application specific parameter. That is, routing decisions are based on this input parameter provided by the client. The destination edge cloud and the path to it is chosen by estimating the response time for all the destination edge clouds and then choosing the minimum among the estimated response times.

5.3 Estimating the Transfer Time on MobilityFirst Virtual Network

The MF architecture provides SETT values which takes into account both bandwidth and latency parameters of the link. Using these parameters along with file or job size submitted by the client, we try to estimate the transfer time as follows:

$$\text{File Delivery ETT} = \frac{\text{file size}}{\min(\text{bandwidth})} + \text{sum of latency}. \tag{5.3}$$

Where, the file size is in *bytes*, bandwidth is in *bytes per second* and latency is in *seconds*. The File Delivery ETT (FDETT) is the estimated transfer time using both bandwidth and latency of the link from client to the edge cloud. We consider minimum bandwidth or minimum throughput in the equation as the link with the lowest bandwidth along the path becomes the bottleneck during the transfer, and latency is taken as the sum of delays on links along the path between the client and the edge cloud.

As mentioned in section 4.3, the flow control mechanism has not been fully optimized in MF, so every chunk of data that is sent across the network is subjected to a sum of delay along the links. When a file transfer is initiated, the file is divided into messages of size 1 MB and these chunks of 1 MB are sent through the network, hence, each chunk of data is subjected to a latency that equals the sum of delays along the path. So we revise equation 5.3 to:

$$\text{File Delivery ETT} = \left\lceil \frac{\text{file size}}{\min(\text{bandwidth})} \right\rceil + \left\lceil \frac{\text{file size}}{\text{message size}} * \text{sum of latency} \right\rceil. \tag{5.4}$$

Where, the file size and the message size are in *bytes*, bandwidth is in *bytes per second*

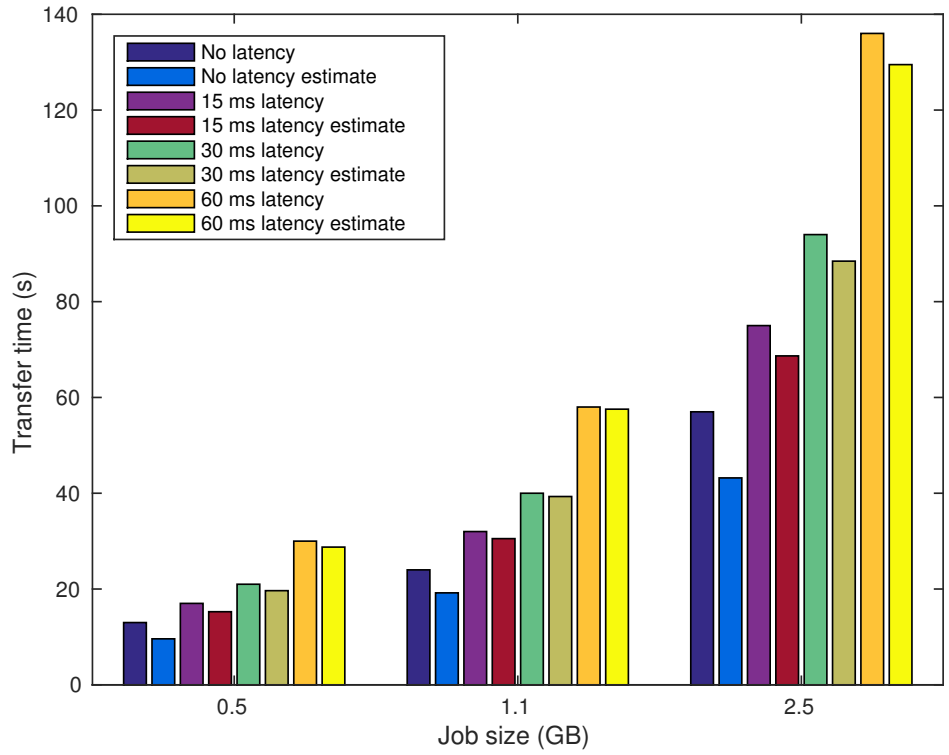

Figure 5.4: Comparison of transfer time and FileDeliveryETT on a 500 Mbps link with varying latency.

and latency is in *seconds*.

Figure 5.4 shows how the estimated values are quite close to the actual transfer times. This estimation is used in equation 5.2 to estimate the response time, and ASR then chooses the best edge cloud to service a client's request.

5.4 Performance of Application Specific Routing

Evaluation of users' QoE for edge cloud services is performed on the ORBIT testbed by replicating simple word count application on multiple edge cloud sites. The edge cloud sites have Hadoop software running on them. Traffic is induced at each of these sites using synthetic traffic generator to control the job completion time, and the link quality is varied using 'netEm' tool. The setup is as shown in figure 5.5 and the block diagram for the same is shown in figure 5.6.

For performance comparison, response time for a client's request is evaluated using

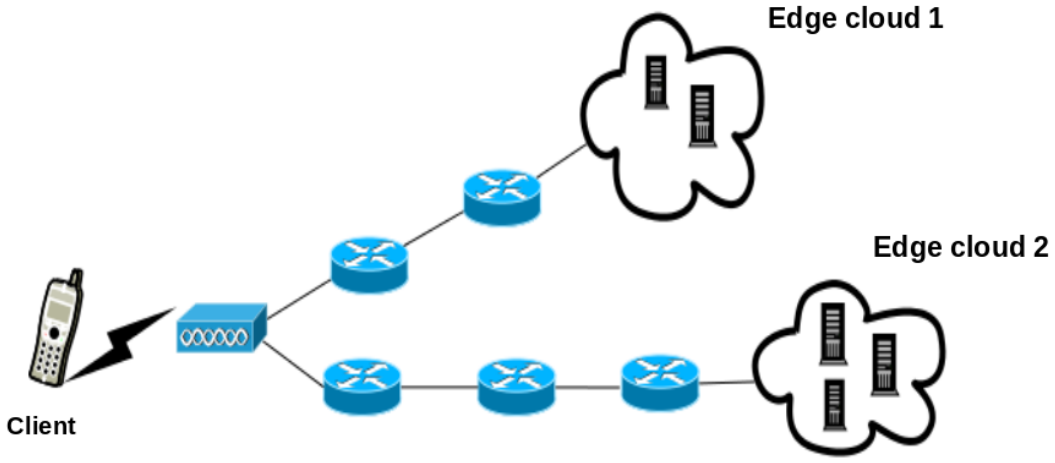

Figure 5.5: Experimental setup to study the performance of ASR against basic anycast service in MF.

ASR and also using the basic anycast service in MF. The basic anycast service in MF chooses the destination that is reachable with the least number of hops, while the ASR algorithm chooses the destination and the path by finding the minimum estimated response time (using equation 5.2) among the available destination nodes. The following tables show the destination edge cloud chosen by ASR and response time for both ASR and anycast service on MF under different link and edge cloud performances.

In the above figures: L1, L2 represent *(bandwidth in Mbps, latency in ms)* from client to the edge clouds 1 and 2 respectively. C1, C2 represent the *average job completion time* in *seconds* at edge clouds 1 and 2 respectively .

It is seen that while anycast always chooses the nearest edge cloud: C1, ASR provides the best response time by either choosing a lightly loaded cluster, or by choosing a high quality link, or a combination of both, and therefore provides a better QoE to the end user.

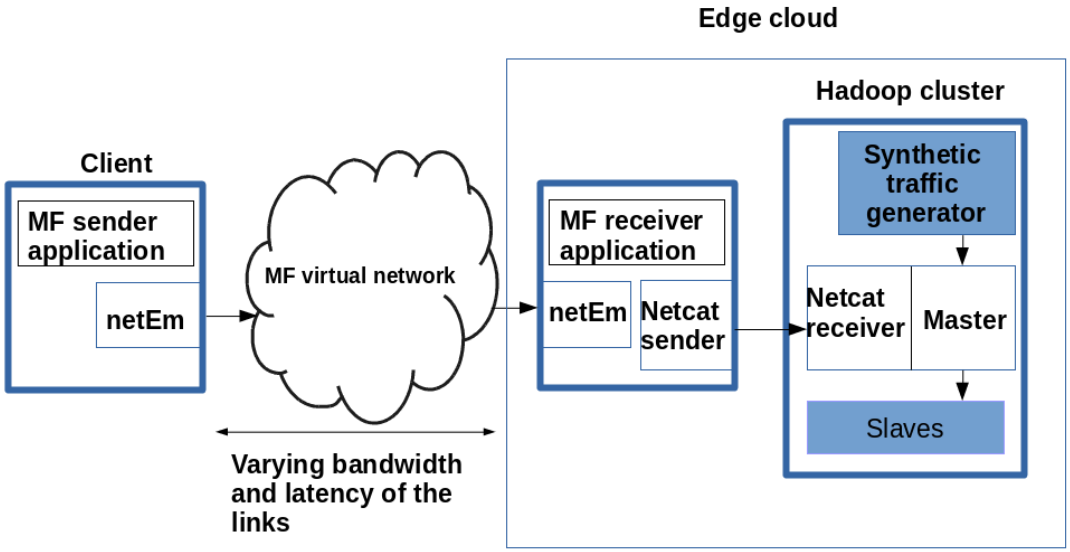

Figure 5.6: Block diagram for the experimental setup in figure 5.5.

File size (GB)	L1	C1	Estimated response time 1	L2	C2	Estimated response time 2	ASR destination
2.5	(100, 30)	200	475	(1000,60)	60	230	C2
1.1	(100, 15)	115	219.5	(100,60)	60	214	C2
2.5	(100, 15)	80	317.5	(100,60)	60	410	C1
0.55	(500,30)	115	140.3	(500,60)	150	191.8	C1
2.5	(500,15)	150	227.5	(500,15)	109	186.5	C2

Figure 5.7: Destination chosen by the ASR algorithm by choosing a minimum of the estimated response times.

File size (GB)	L1	C1	L2	C2	ASR destination	Anycast destination	ASR response time	Anycast response time
2.5	(100, 60)	200	(1000,15)	60	C2	C1	251.7	828.36
1.1	(100, 15)	115	(100,60)	60	C2	C1	253.5	334.38
2.5	(100, 15)	80	(100,60)	60	C1	C1	443.23	443.23
0.55	(500,30)	115	(500,60)	150	C1	C1	127.41	127.41
2.5	(500,15)	150	(500,15)	109	C2	C1	292.45	412.4

Figure 5.8: Performance comparison of ASR against basic anycast service in MF.

Chapter 6

Future Work

6.1 Service migration

For mobile users, service migration is one of the techniques that is used to keep the edge cloud, servicing a user's request in close proximity to the user in order to reduce latency [23]. Service migration involves the process of transferring application data from one edge cloud to another.

Let's say a user makes a request at an edge cloud site A and moves. It is now in close proximity to edge cloud site B. Is there a need for service migration from A to B by transferring the state or would A still provide the user with good QoE? By knowing the size of the state that needs to be transferred, link performance from A to B and edge cloud performance of B, we can use ASR in deciding whether there is a need to migrate. Service migration always comes with a certain cost and would be best if large state transfers can be avoided. ASR can support the application in deciding whether it is necessary to migrate and thus avoids unnecessary costs to ensure user's QoE. Further work needs to be done in designing a module that is supported by ASR that can support service migration.

6.2 Edge cloud performance metric

Currently, we use utilization or traffic intensity at the edge cloud site as a parameter to measure the edge cloud performance which contributes to the ASR metric. This utilization factor is calculated over a long period of time and it reflects the performance history of the system rather than the current performance which may provide better

results. Therefore, there is a need for edge cloud performance metric to reflect the current workload or most recent average job completion time by measuring the utilization of the system or job completion time over a particular period of time or a window size. It may also be necessary to take into account variations in the job arrival process and the specific scheduling policies used at the cluster.

References

[1] Fog Computing: Extending the Cloud to the Edge, http://www.datacenterknowledge.com/archives/2013/08/23/welcome-to-the-fog-a-new-type-of-distributed-computing/.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[3] I. Seskar, K. Nagaraja, S. Nelson, and D. Raychaudhuri, "MobilityFirst Future Internet Architecture Project," in *MobilityFirst Project, Proc. ACM AINTec 2011*.

[4] NSF Future Internet Architecture Project, http://www.nets-fia.net/.

[5] T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, and D. Raychaudhuri, "Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 698–707.

[6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[7] S. Velrajan, "Application aware routing in software defined networks."

[8] "Mobilityfirst future internet architecture project," http://mobilityfirst.winlab.rutgers.edu/.

[9] S. C. Nelson, G. Bhanage, and D. Raychaudhuri, "GSTAR: Generalized storage-aware routing for MobilityFirst in the future mobile Internet," in *Proceedings of MobiArch'11*, 2011, pp. 19–24.

[10] A. Riedl and D. A. Schupke, "A flow-based approach for ip traffic engineering utilizing routing protocols with multiple metric types," in *Proceedings of the 6th INFORMS Telecommunication Conference, Boca Raton, USA*, 2002.

[11] Introduction to EIGRP - Cisco, http://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/13669-1.html.

[12] Design and implementation of a virtual networking framework for the MobilityFirst Future Internet Architecture, http://www.winlab.rutgers.edu/ ababu/downloads/thesis.pdf.

[13] Apache Hadoop, https://hadoop.apache.org/.

[14] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.

[15] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, "Map-reduce-merge: simplified relational data processing on large clusters," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 1029–1040.

[16] B. T. Rao, N. Sridevi, V. K. Reddy, and L. Reddy, "Performance issues of heterogeneous hadoop clusters in cloud computing," *arXiv preprint arXiv:1207.0894*, 2012.

[17] S. Hemminger *et al.*, "Network emulation with netem," in *Linux conf au*. Citeseer, 2005, pp. 18–23.

[18] T. Kelly, "Scalable tcp: Improving performance in highspeed wide area networks," *ACM SIGCOMM computer communication Review*, vol. 33, no. 2, pp. 83–91, 2003.

[19] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[20] TCP Receive Window Auto-Tuning, https://technet.microsoft.com/en-us/magazine/2007.01.cableguy.aspx.

[21] TCP Extensions for High Performance, https://www.ietf.org/rfc/rfc1323.txt.

[22] WINLAB, ORBIT, http://www.orbit-lab.org/wiki/WikiStart.

[23] D. Arora, M. Bienkowski, A. Feldmann, G. Schaffrath, and S. Schmid, "Online strategies for intra and inter provider service migration in virtual networks," in *Proceedings of the 5th International Conference on Principles, Systems and Applications of IP Telecommunications*. ACM, 2011, p. 10.