International Journal of Scientific Engineering and Science
Volume 6, Issue 11, pp. 24-29, 2022.
ISSN (Online): 2456-7361

Integration of Performance Isolation, Bandwidth Management and Burst Handling for QOS Management in IP SANs

[1]Joseph Kithinji, [2]Makau S. Mutua, [3]Gitonga D. Mwathi
[1]Department of Computer Science, Meru University of Science and Technology, Meru, Kenya
E-mail: joskithinji2014@gmail.com
[2]Department of Computer Science, Meru University of Science and Technology, Meru, Kenya
E-mail:smutua@must.ac.ke
[3]Department of Computer Science, Chuka University, Chuka Kenya
E-mail: dgmwathi@chuka.ac.ke

Abstract— *The increasing number of Information technology Users around the world has led to tremendous increase in the amount of data that requires storage. In response to this challenge, new storage area network architectures based on Ethernet (IP) have evolved. With the coexistence of storage traffic with other types of traffic in the same IP network, it is important to offer storage traffic QOS guarantees to prevent performance degradation for storage users. Regrettably, the storage device itself does not provide any capability of guaranteeing storage QOS.Commercially available storage arrays offer only limited support providing performance isolation bandwidth management and burst handling. To address this issue this paper introduces an integrated solution that enforces performance isolation, bandwidth management and burst handling among IP SANs.Through experiments it was established that the proposed solution is able to take advantage of its knowledge of traffic patterns to adjusts priorities to provide QOS(Quality of service) close to the service level objective(SLO) of storage users.*

Keywords— *Service level objective,Quality of Service,IP SANs,performance isolation,bandwidth management,traffic shaping.*

I. Related Works

With the advent of ISCSI the IP network is able to accommodate the transmission for storage data. This means that storage traffic and other types of traffic will mix. This brings about a challenge of providing QOS to the vast variety of traffic flow requirements. TCP provides best effort which is unsatisfactory for providing QOS to storage traffic. Providing QOS guarantee require a number of functions to be performed such as performance isolation, bandwidth management and traffic shaping. There has been many proposed solutions for providing QOS in IP SANs.

Jaiman et al.2019) developed Heron which is an algorithm that is aimed at reducing tail latency when dealing with heterogeneous workloads. However this technique relies on predictions. If the predictions are wrong then resources may be wasted.

Peng et al., (2019) Developed fair-EDF to provide latency guarantees for storage servers. Results obtained showed that fair-EDF is able to provide fairness for heterogeneous workloads. However this mechanism was found not to be scalable. In addition fair-EDF lacks the mechanism to separate workloads with large execution time from those with small execution time.

Peng et al., (2019) also developed pShift which is a token allocation algorithm for balancing resources between storage servers. However it was found to have scalability problems. Peng & Varman, (2018) came up with Bqueue which is a scheduling system that provides QOS reservations and limits. To handle dynamic workloads bandwidth is computed at regular intervals. The problem with Bqueue was found to be that it uses tokens allocation as the only control measure for QOS.

Cui et al.,(2019) developed tail cutter as a mechanism for reducing latency in cloud storage systems. Tailcutter uses parallel request to cloud datacenters to reduce latency. However it only uses latency as a QOS control measure for storage.

Peng & Varman(2020) Developed pTrans which is a framework for reservation allocation based on direct graph model. However pTrans is not able to give accurate estimates for resources available at run time which leads to inaccurate allocation of resources and wastage. In addition pTrans was found to increase with workload.

Techniques like PARDA(Gulati & Waldspurger, 2009.), Argon(Wachs et al., 2007) use queue length management and disk time reservation for implementing proportional throughput fairness as a means for implementing QOS in storage area networks. Other techniques like

Technique such as Priority Meister (Zhu, Tumanov, Kozuch, & Ganger, 2017) and Triage(Karlsson, Karamanolis, & Zhu, 2005) use throughput performance isolation among competing workloads by use I/O throttling techniques such as Leakey bucket algorithim, deficiet round robin and start-time fair queuing(SFQ) to manage how much throughput competing workloads receive. Other techniques like mClock(Gulati & Varman, 2007) and Pisces(Shue, Freedman, & Shaikh, 2012) support throughput QOS using maximum minimum fairness.

Other studies like those done in cosTLO(Wu, Yu, & Madhyastha, 2015) and C3 (Suresh et al., 2015), use redundancy to reduce coverage and tail latencies.C3(Suresh et al., 2015) reduces tail latency through dynamic redundancy and distributed rate control.

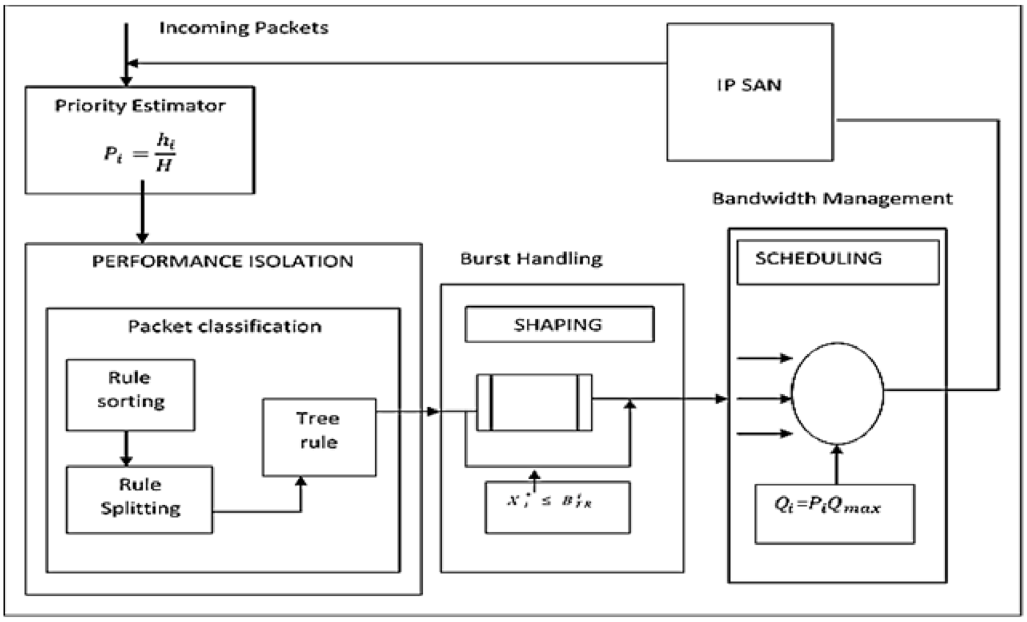

Fig. 1. Integrated QOS Management Technique Architecture.

Previous works reviewed in this research includes techniques only either for latency support and only those for throughput support. In contract this research implements an integration of three techniques in an attempt to support throughput, latency, and jitter for users in IP SANs.

To improve on the previous work this study integrates the three functions of performance isolation, bandwidth management and burst handling otherwise used separately in the previous studies. These integration is aimed at increasing throughput, reducing latency and reducing jitter

In addition most of the techniques reviewed in chapter one are either predictive, static or do not take into consideration the network statistics. To further improve on the previous work this paper incorporates the features of dynamism, use of network statistics to prioritize traffic and finally the use of a centralized mechanism to reduce the overhead experienced when multiple copies of the same algorithm are run. To measure the performance of the proposed system in providing QOS the metrics of throughput, latency, and jitter are used.Fig.1. Illustrates the proposed system architecture.

Priority Estimation Module

The priority estimation module is designed to capture the current network statistics and calculate the priority of each flows *i* .The priority is meant to be used to forecast on the amount of resources a certain class of users requires. For each flow *i* the priority estimation module calculates its priority using equation $p_{i=\frac{h_i}{H}}$.

Performance Isolation Module

A packet d_i is classified based on the header information that is source IP, destination IP, Source port, destination port and protocol. The same fields are components of rule r_i. A packet d_i is said to match rule r_i if and only if $di.fi{=}{=}r_i.f_i$.Based on the rule that match the associated action is performed. After classification packets are forwarded to the shaper

Burst Handling Module

The burst handling module is meant to delay packets so that they form a constant flow. Burst handling is implemented using traffic shaping. The proposed traffic shaping algorithm takes in various QOS classes *i (i=1...n)* and uses a dynamic time interval t_s to send traffic in burst .Each session consist of *n* queues q_i each containing flows belonging to the same class and priority. High priority queue are placed at the top .At any time if the queue is not full and the time t_s has not elapsed the incoming flows are not delayed. Otherwise the packets are sent and the t_s is reset to zero. The two events that trigger the sending of packets is when $X_i^* \leq B_{TR}^i$and t_s has expired.

Bandwidth Manager

The bandwidth management algorithm begins by establishing the quantum Q_i which is the amount bits that can be transmitted in each round from queue i based on priority $p_i = \frac{h_i}{H}.Q_i$ Represents the value$P_i\, XQ_{Max}$. Where Q_{Max} is the maximum possible size of any packet that can exist in the network. To ensure service differentiation queues are arranged hierarchically in one level instead of one FIFO queue found in best effort.

II. VALIDATION OF TECHNIQUE

Validation Metrics

Experiments were set up to establish the performance of the proposed system based on the QOS metrics of throughput, jitter and latency. Reads were simulated to mimic the real IP SAN environment.

(1) and (2) were used calculate the percentage throughput deviation and latency deviation respectively

$$\%throughput\ deviation = \frac{attained\ throughput - expected\ throughput}{expected\ throughput} X100 \quad (1)$$

$$Latency\ deviation = observed\ latency - expected \quad (2)$$

The following sections presents the results throughput, latency and jitter obtained by using I/O sizes of 4KB, 64KB and 1MB for a period of 200 seconds. For all the experiments three scenarios were considered corresponding to IO sizes of 4KB, 64KB and 1MB.

TABLE 1: SLO for Classes of Storage Users

Class of user	IOPS	Throughput for Block size 4KB	Throughput Block size 64KB	Throughput Block size 1MB	Response time for QD32
Task user	5 IOPS	20kb/s	320kb/s	5000kb/s	6.4 ms
Knowledge user	10-20 IOPS	40-80kbs	640-1280kbs	10240-20480kb/s	2.4 ms
Power user	25 IOPS	100kb/s	1600kb/s	25000kb/s	1.3 ms

User QOS Mapping

Different users have varied QOS requirements which should be matched to corresponding QOS requirements. Users flows mapped to the same SLO are put on the same queue. Through this mapping the router can be able to provide differentiated treatment of flows. The IT world classifies storage users into three main classes that is the power user, knowledge users and task users. Based on the user's requirements in delay and throughput we map users to three QOS levels. The mapping relations are shown in Table 1. Power users and knowledge users are sensitive to delay as illustrated by the low latency/response time. The task users are less sensitive to delay however they require bandwidth guarantee.

Therefore from Table 1 the SLO for users based on the IOPs and block size was derived. The values for the SLO are throughput in kb/s followed by IOPS and latency. For a block size of 4KB the SLO for task, knowledge and power users is as follows; task users(20kb/s,5IOPS,6.4MS), Knowledge users (60kb/s,15IOPS,1.6-3.2ms) and power users (100kb/s,25IOPS,1.3ms). The same case applies for 64kb and 1 Mb block sizes.

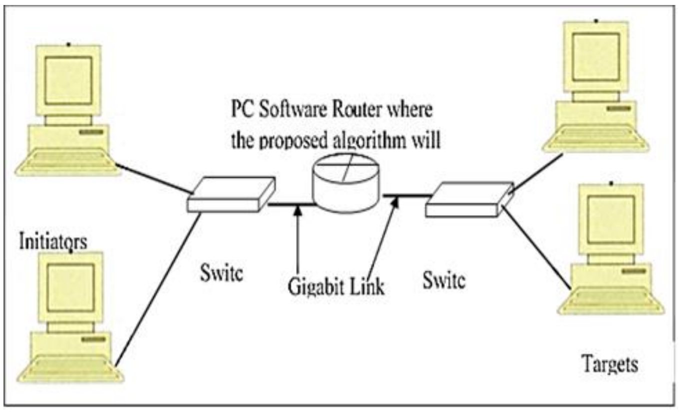

Fig. 2. Experimental Setup for the proposed system

Validation Setup

The validation experiment is as illustrated in Figure 2. Park dale disk benchmarking tool was used to simulate the reads and writes. The initiators were setup with initiator mode ISCSI driver while the target storage were configured target mode. Experiments were used to validate the proposed systems based on latency, throughput and jitter. In all the experiments a File size of 50MB was used unless otherwise stated. All experiments were run three times for a period 200 seconds and averages recorded.Fig.2 illustrates the experimental setup.

III. Validation Results

Throughput versus IO Size

Fig.3 shows the throughput of task users, knowledge users and power in the best effort case and using the proposed solution scenarios. In best effort scenario in as illustrated in Fig.3(a), the lack of QOS management scheme causes hosts to have unstable throughput and a lot of unfairness. Ideally power users would perform better than other users. In the proposed solution scenario as illustrated by Fig.3(b), the unfairness is corrected by isolating users by decoupling the throughput of the three classes of users and lets them process packets at their own rates. Generally from Fig.3(b) it is observed that at *t=0* there is the lowest throughput which increase up to *t=20* where it stabilizes. The stability is brought about by the proposed solution being able to optimize bandwidth usage as well as isolate performance of one flow from the other. Table 2 further illustrates the results of scenario 1.

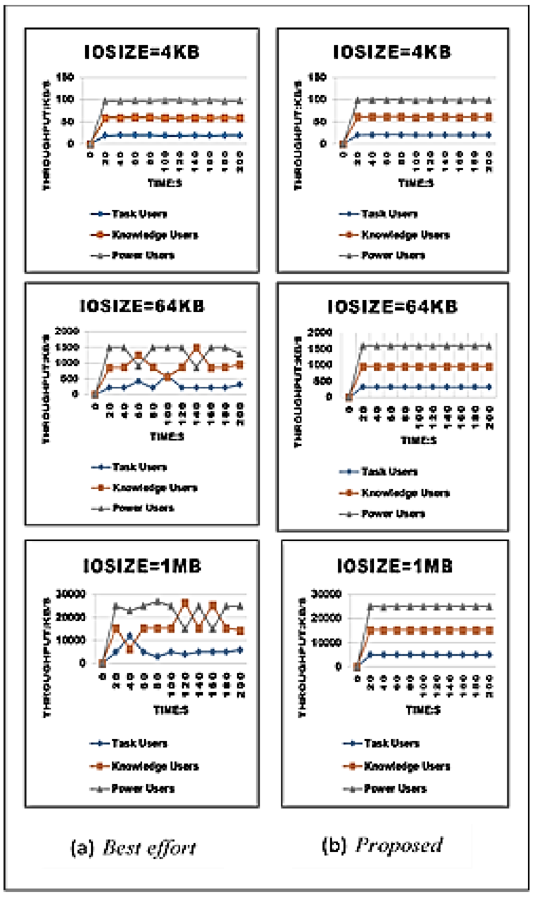

Fig. 3. Throughput for 200 seconds ;(a) Best effort (b)Proposed Solution.

Scenario1 represents the situation when using an IO size of 4KB for both using the proposed solution and best effort. From Table 2 it is evident that all users receive a throughput close to the SLO with a negative percentage deviation from the SLO of 3%, 0.8% and 0.2 Kb/s for task users, knowledge users and power users respectively when using the proposed solution. The same is observed when using the best effort. That is the task users, knowledge users, and power users attained a negative percentage deviation of 5%, 1.6% and 1%. This can be explained since with 4KB IO size

TABLE 2: Scenario 1 with IO size of 4KB

Class of User	Expected SLO Throughput	Proposed Solution Throughput	Existing solution	Deviation 1	Deviation 2
Task users	20	19.4	19	0.6	1
Knowledge users	60	59.5	59	0.5	1
Power Users	100	99.8	99	0.2	1

TABLE 3: Scenario 2 with IO size of 64KB

Class of user	Expected SLO Throughput	Proposed Solution Throughput	Best Effort	Deviation 1	Deviation 2
Task users	320	300.16	263.13	19.84	56.87
Knowledge users	960	919.55	858.09	40.45	101.91
Power Users	1600	1540.03	1282.22	59.97	317.78

TABLE 4: Scenario 3 with IO size of 1MB

Class of user	Expected SLO Throughput	Proposed Solution Throughput	Best Effort	Deviation 1	Deviation 2
Task users	5000	4012.3	3543.30	987.7	1456.7
Knowledge users	15000	12750	11762	2250	3238
Power Users	25000	22890	20896	2110	4104

Generally from Fig.3(b) it is observed that at *t=0* there is the lowest throughput which increase up to *t=20* where it stabilizes. The stability is brought about by the proposed solution being able to optimize bandwidth usage as well as isolate performance of one flow from the other. Table 2 further illustrates the results of scenario 1.

In scenario 2 when using an IO size of 64KB the following observation were made. With the proposed solution there is a negative percentage deviation from the SLO of 6.2%, 4.2% and 3.7% for task users, knowledge users and power users respectively .On the other hand when using best effort a negative percentage deviation from the SLO of 17%, 10.6% and 19.86% for task users, knowledge users and power users respectively .Table 3 illustrates scenario 2.

This can be explained by the fact that an increase in IO size results in a corresponding increase in traffic which causes congestion(Jaiman et al., 2018). However for the proposed solution since it implements performance isolation, bandwidth management and traffic shaping the deviation is minimal compared to that of best effort. Table 4 represents the results of scenario 3.

In scenario 3 when using an IO size of 1MB the following observation were made. With the proposed solution there is a negative percentage deviation from the SLO of 19.8%, 15% and 8% for task users, knowledge users and power users respectively .On the other hand when using best effort a negative percentage deviation from the SLO of 29.1%, 22% and 16% for task users, knowledge users and power users respectively .

This can be explained by the fact that an increase in IO size results in a corresponding increase in traffic which causes congestion(Jaiman et al., 2018). However for the proposed solution since it implements performance isolation, bandwidth management and traffic shaping the deviation is minimal compared to that of best effort.

It is further observed that when using best effort the task user's experiences the greatest deviation for scenario 2 and scenario 3 and the lowest is experienced by knowledge users. This is contrary to what is expected given that power users have got higher priority and therefore should have a smaller percentage deviation. This can be explained by the fact that best effort technique lacks the mechanism of prioritization present in the proposed solution. This is consistent with results obtained in (Gulati & Varman, 2010) where it was found that resource reservations and controls are able to provide predictable performance. In the results obtained the expectations were that high priority users should be provided with predictable service. The results obtained were consistent with the expectations and those obtained by Billaud and Gulati(2013),Gulati and Waldspurger(2009) and Peng(2019) proving that the proposed algorithm is work conserving

Latency and IO size

Latency is the time it takes for a packet to reach its destination. Latency has a lot of effect on network performance degradation and effects the user QOS. High latency is caused congestion which results in poor QOS. In this case the study considered end to end delay that is the time taken from source to destination(Jaiman et al., 2019). Fig.4 analyzes the latency experienced by the three classes of users against time for best effort and proposed solution. Latency was measured for three scenarios for IO sizes of 4KB, 64KB and 1MB.

For scenario 1 where there is an IO size of 4KB it was observed a lower that all users experienced a latency lower than that expected for both the proposed solution and the best effort. Even though the best effort has no QOS mechanisms implemented here in the proposed solution, all the users still meet their deadlines. This can be explained by the fact that when a small IO size is small there is low congestion since they occupy the network for a short time(Jaiman et al., 2019) which does not lead to resource competition and therefore does not require any management.

TABLE 5: Scenario 1 with IO size of 4KB

Class of User	Expected SLO Latency	Proposed Solution Latency	Best Effort Latency	Deviation 1	Deviation 2
Task users	6.4	5.6	6.0	-0.8	-0.4
Knowledge users	3.2	2.4	2.9	-0.8	-0.3
Power Users	1.3	0.6	1	-0.7	-0.3

TABLE 6: Scenario 2 with IO size of 64KB

Class of user	Expected SLO Latency	Proposed Solution Throughput	Best Effort Latency	Deviation 1	Deviation 2
Task users	6.4	5.8	7.7	-0.6	+1.3
Knowledge users	3.2	2.8	5.4	-0.4	+2.2
Power Users	1.3	1.1	2.6	-0.2	+1.3

TABLE 7: Scenario 3 with IO size of 1MB

Class of user	Expected SLO Latency	Proposed Solution Throughput	Best Effort Latency	Deviation 1	Deviation 2
Task users	6.4	6.0	10.79	-0.4	+4.39
Knowledge users	3.2	2.8	10.47	-0.4	+7.27
Power Users	1.3	1.1	5.52	-0.2	+4.22

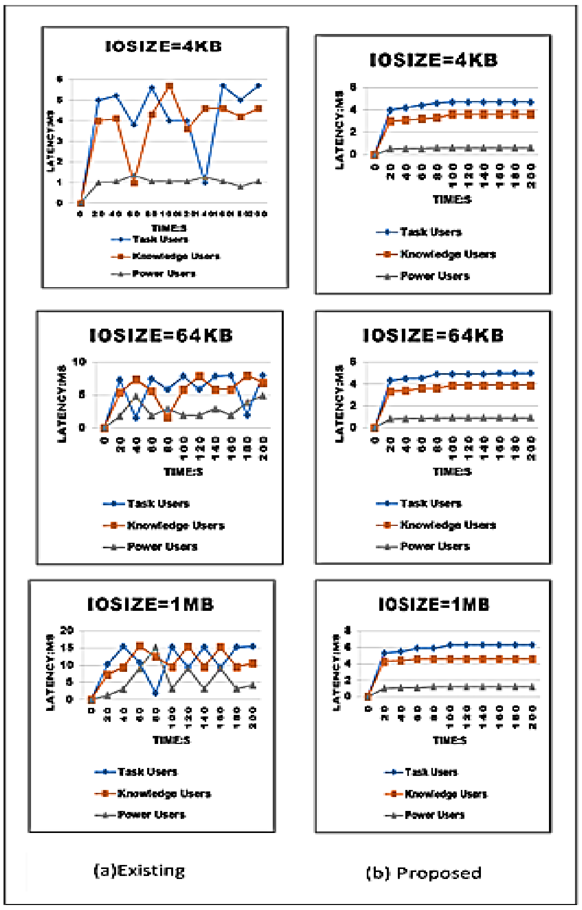

Fig. 4. Latency for 200 seconds (a) Best effort, (b) Proposed solution

In scenario two and three an IO size of 64KB and 1MB are used respectively. An increase in IO size resulted in an increase in the traffic which lead to competition of bandwidth(Jaiman et al., 2019). As a result as depicted in Table 5 ,6 and 7 where it shows that for the proposed solution a negative deviation was achieved for all users .A negative deviation means that users were able to achieve a latency lower than expected which means they were able to meet their deadlines. Conversely with best effort, it was observed that all users attained a positive deviation which means that user surpassed the latency threshold that was expected and none met their deadlines. This phenomena can be explained by the fact that best effort lacks the QOS techniques of performance isolation, bandwidth management and burst handling implemented in the proposed solution.

Absence of these mentioned techniques results in free for all competition for bandwidth due to lack of prioritization mechanisms users are able to interfere with each other resulting in ununiformed latency(Gulati & Varman, 2007). In addition FIFO queues used in best effort do not provide a way for isolating traffic. An increase in latency also can be attributed to the use of DRR in best effort. When using DRR for scheduling, big packets lead to an increase in head of line latency which delays smaller maybe more important packets. Results obtained by Wang et al., (2012) also showed similar pattern where it was found that achieving low latency requires smaller queues. Lack of optimized scheduling algorithm results in larger queues which results in increased latency as witnessed in best effort. Similarly mixing of big and small packets results in headline delay causing more latency for smaller packets as witnessed when best effort is used. It is further noted that for all the scenarios all users experience a low latency between time t=0 and time *t=20*. This is because before the 20 seconds traffic has not reached saturation therefore there is less latency. In conclusion it is important to note that when using the proposed solution for each class of user the latency goals are satisfied contrary to when using best effort. When using best effort latency increases by a factor of 2X.This demonstrates the inability of conventional scheduling techniques in providing acceptable latencies in presence of huge traffic loads.

Jitter and IO size

Jitter is the variation in delay experienced by packets reaching a destination thus its presence is unwanted but unavoidable. Therefore there is always a small amount of jitter. From the experiments jitter observed under the proposed solution and best effort was recorded as shown in Table 7 and Fig.5.

Fig.5 analyzes the jitter experienced for 200 seconds for IO sizes of 4kb, 64KB and 1MB. At t=0 a jitter of 0 was observed and therefore the system experiences the best performance at t=0.

From Fig.5 it is observed that when an IO size of 4kbyte is used that there is no congestion and therefore jitter of the three classes of users is small.

This is consistent with results obtained by Peng and Varman (2020) and Jaiman et al., (2019) that the larger the IO size the more time the traffic occupies the network and therefore the more the jitter.

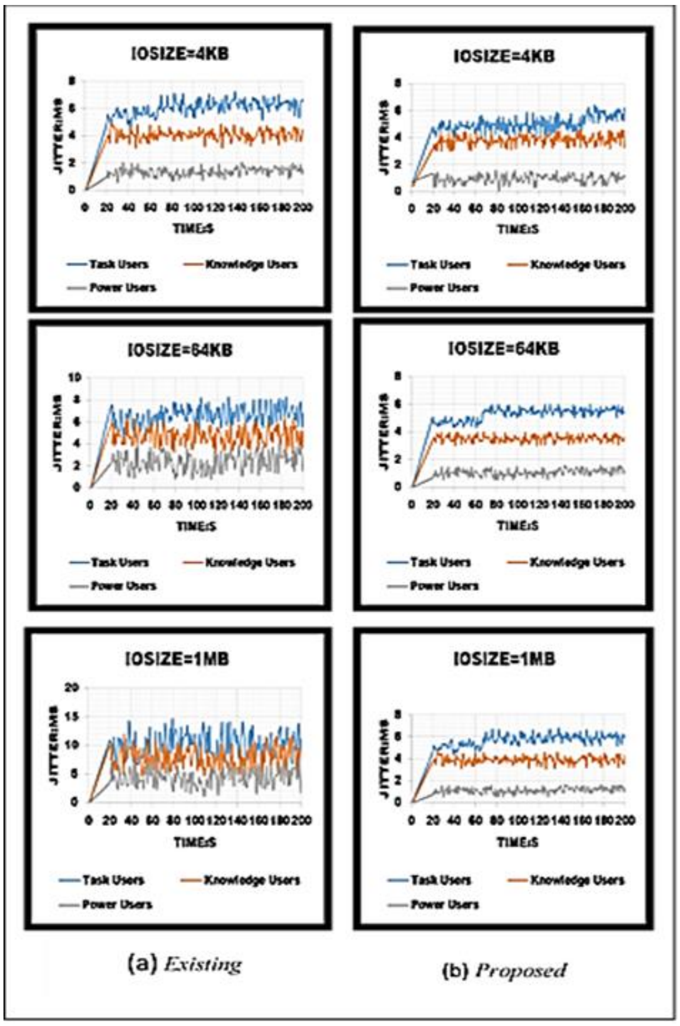

Fig. 5. Jitter for 200 seconds (a) Existing (b) Proposed Solution

For an IO size of 64KB the best effort technique a jitter of 6.5 for task users, 4.7 for knowledge users and 2.3 for power users was observed. While for the proposed system jitter of 5.2, 3.5 and 1.1 were observed for task users, knowledge users and power users respectively. A reduction of 20%, 25% and 52% on average. This clearly shows that the proposed system outperforms the best effort. The same trend of an increase in jitter is observed for 1MB. The reason why the proposed technique outperforms the best effort is that the proposed technique uses a hierarchy of levels for flows which isolates traffic and avoids interference between flows as opposed at the best effort where all flows are using single FIFO queues. Minimum jitter was observed for 4kbyte, while maximum jitter was observed for 1MB IO size. From the results it is also observed that task users have the highest jitter for the given configuration. This can be attributed to their low priority.

TABLE 8: Average Jitter in Milliseconds

		Proposed Solution			Existing		
Storage user	**Time**	**4KB**	**64KB**	**1MB**	**4KB**	**64KB**	**1MB**
Task users	200s	4.2	5.2	5.6	6.0	6.5	9.9
Knowledge users	200s	3.2	3.5	3.8	4.0	4.7	7.6
Power users	200s	1.0	1.1	1.2	1.3	2.3	4.4

IV. SUMMARY

In this paper the integrated approach has been implemented that includes the QOS techniques of performance isolation bandwidth management and traffic shaping. The performance isolation module ensures that flows don't interfere with each other performance. The bandwidth management module ensures that each flow/class of user gets a share proportion to its current need. This is achieved through regular computation of priority. The proposed algorithm is implemented in Linux router and causes little delay. Through experimentation it has been verified that the proposed solution works as intended.

REFERENCES

1. Cui, Y., Dai, N., Lai, Z., Li, M., Li, Z., Hu, Y., … Chen, Y. (2019). TailCutter: Wisely cutting tail latency in cloud CDNs under cost constraints. *IEEE/ACM Transactions on Networking*, *27*(4), 1612–1628. https://doi.org/10.1109/TNET.2019.2926142
2. Gulati, A., Ahmad, I., & Waldspurger, C. A. (2009, February). PARDA: Proportional Allocation of Resources for Distributed Storage Access. In *FAST* (Vol. 9, pp. 85-98).
3. Gulati, A., Merchant, A., & Varman, P. J. (2007). pClock: an arrival curve based approach for QoS guarantees in shared storage systems. *ACM SIGMETRICS Performance Evaluation Review*, *35*(1), 13-24.
4. Gulati, A., Shanmuganathan, G., Zhang, X., & Varman, P. (2019). Demand based hierarchical QoS using storage resource pools. *Proceedings of the 2012 USENIX Annual Technical Conference, USENIX ATC 2012*, 1–13
5. Karlsson, M., Karamanolis, C., & Zhu, X. (2005). Triage: Performance Differentiation for Storage Systems Using Adaptive Control. *ACM Transactions on Storage*, *1*(4), 457–480. https://doi.org/10.1145/1111609.1111612
6. Peng, Y. (2019). Latency Fairness Scheduling for Shared Storage Systems. *2019 IEEE International Conference on Networking, Architecture and Storage (NAS)*, 1–8.
7. Peng, Y., Liu, Q., & Varman, P. (2019). Scalable QoS for Distributed Storage Clusters using Dynamic Token Allocation. *IEEE Symposium on Mass Storage Systems and Technologies*, *2019-May*, 14–27. https://doi.org/10.1109/MSST.2019.00-19
8. Peng, Y., & Varman, P. (2018). BQueue: A coarse-grained bucket QoS scheduler. *Proceedings - 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2018*, 93–102. https://doi.org/10.1109/CCGRID.2018.00024
9. Peng, Y., & Varman, P. (2020). PTrans: A Scalable Algorithm for Reservation Guarantees in Distributed Systems. *Annual ACM Symposium on Parallelism in Algorithms and Architectures*, 441–452. https://doi.org/10.1145/3350755.3400273
10. Wachs, M., Abd-El-Malek, M., Thereska, E., & Ganger, G. R. (2007, February). Argon: Performance Insulation for Shared Storage Servers. In *FAST* (Vol. 7, pp. 5-5).