HPDDRR: Optimized Scheduler Shaper for Bandwidth Management and Traffic Shaping in Internet Protocol Storage Area Networks

Kithinji Joseph
Computer Science Department,
Meru University of Science and Technology
Meru, Kenya

Makau S. Mutua, PhD
Computer Science Department,
Meru University of Science and Technology
Meru, Kenya

Gitonga D. Mwathi, PhD
Department of Computer Science,
Chuka University
Chuka, Kenya

ABSTRACT
Providing QOS (quality of service) is a vital problem in storage area networks. In this paper a technique known as HPDDRR(hierarchical priority based dynamic deficit round robin) which is scheduler shaper that uses hit ration for flow prioritization and a dynamic quantum calculated based on the priority for scheduling is presented. Based on the applications used, packets may vary in sizes and belonging to different priority classes. To ensure that big low priority packets don't delay small high priority packets this study uses hierarchical priority queues instead of FIFO (first in first out) queues for scheduling. This allows for performance isolation as well as resource sharing. The evaluation results proof that HPDDRR is able to optimize bandwidth utilization as well as latency for competing traffic flows under Service level objectives constraints.

Keywords
Dynamic Bandwidth management, Burst Handling, ISCSI, IP SAN, Quantum, Policing.

1. INTRODUCTION
With the ever increasing demand for storage, IP SANs (Internet protocol storage area networks) are becoming popular option due to operational and hardware cost savings[1][4]. An IP SAN is a storage area network that uses the ISCSI(internet small computer system interface) to transport storage data in a network[2].ISCSI is an internet protocol based standard for transporting storage commands over the IP network. The reads and writes are encapsulated in ISCSI then transported through the TCP/IP (Transmission control/Internet Protocol) network. However the TCP/IP does not provide mechanism for regulating the bandwidth allocated to a particular user[4][8]. In addition, compared to other SANs, in IP SANs the storage traffic mixes with other types of traffic. These presents a new challenge and opportunity for bandwidth management for storage users. The challenge is network traffic as well as storage traffic are bursty, and therefore a mechanism for managing the link is required. The opportunity is that it's possible to adopt the existing bandwidth management techniques developed over the years for data network to IP SANs[3][7].

IP SANs have two resources that need to be managed, that is the storage and the network. In this study the resource under contention and which requires management is the bandwidth between the initiator and the target. The amount of available bandwidth determines the amount of data that can be transmitted[5]. This amount of data is known as throughput[6]. In the traditional network setup the amount of bandwidth is fixed and provides best effort which does not provide any resources guarantees to any users on the network[3]. Future traffic patterns are unknown which makes bandwidth management and burst handling a challenge. For efficient use of IP SAN the network bandwidth among the clients must be distributed dynamically depending on the client's workload.QOS solutions such as interserve, diffserve, RSVP are not effective while applied directly in the storage system[9].

Dynamic bandwidth management is the ability of a bandwidth management scheme to adjust bandwidth allocations based on network conditions[10][3].In most routers we have two main algorithms for dynamic bandwidth management that is the per connection queue (PCQ) and hierarchical token bucket (HTB)[11].PCQ is a non priority class based queuing algorithm used to throttle bandwidth. Due to the lack of prioritization PCQ is not able to differentiate traffic. For this case of IP SANs it is important to provide better services for high hitting classes which are assigned higher priority. Because of these fact HTB[9] is used. However HTB uses DRR scheduling mechanism.[15]DRR is known to have high latency and also leads to low bandwidth utilization of resources especially when there are flows in the same queue with different rates[12].

On the other hand burst handling is implemented using traffic shaping. Traffic shaping is a congestion control technique that delays traffic of less important classes in an attempt to optimize network performance[13]. This is done by limiting the burst size so that it does not exceed the network limit. Two architectures are available for traffic shaping. These include; direct exact sorting and rate based grouping[14]. Direct exact sorting operate on the basis of per virtual connection queue at the input port. After the per virtual connection queue there are timing queues which are formed considering the incoming flows departure time[11]. The shortcomings of these technique is that the implementation complexity increases linearly as the number of connections increases. This follows from the fact that the complexity of direct exact sorting architectures is O (p_{max}/p_{min}).

On the other hand rate based architecture for traffic shaping allows for grouping of traffic based on rates, however the groups become many when the number of connections increases[18]. This increases the complexity and delays in packet processing. In addition both direct and rate based techniques for traffic shaping employs the FIFO queues which makes it difficult to differentiate traffic[16].

To solve the above mentioned problem on bandwidth management and traffic shaping the study adopts a

scheduler/shaper named hierarchical priority based dynamic deficit round robin (HPDDRR) that employs the technique of hierarchy structure of flows to reduce the number classes queues, uses priority calculated from hit ratio of flows to calculate the deficit quantum which ensures that the quantum is dynamic based on network statistics. The proposed solution uses a hierarchy to queue packets instead of the FIFO queues[17]. The hierarchical structure allows for isolation of traffic between flows. In addition so as to retain the complexity of O (1), the hierarchical structure will have one level[14]. The complexity of a hierarchical structure was found to be O (L).Where L is the number of levels of hierarchy structure. Consequently proposed solution implements only one level hence retaining O (1) complexity of DRR.This is expected to improve on latency compared to the conventional rate based scheduler shaper using the conventional DRR[19].The property of dynamic counter is meant to ensure packets get transmitted as much as possible in every round robin as the deficit will be calculated based on highest rate of the highest priority queue. Thisis expected to improve on bandwidth utilization since a class will be allocated bandwidth based on the current network requirements. The feature of traffic classification further improves on latency as packets of similar rates are grouped in the same queue which reduces the waiting time which might be high for low rate packets when mixed with high rate packets.

2. PREVIOUS WORK

[20] QOS in SANS have been researched for years with solutions such as Façade, chameleon, triage and Stonehenge being proposed. Facade uses the technique of throttling I/O requests to the storage to achieve the required SLO.However façade earliest deadline is not effective when we have burst workloads. Chameleon leaky bucket is not efficient since it is not work conserving as it reserves bandwidth to support each client's storage QOS requirements sharing of resources proportionally. Solutions such as YFQ and cello balance user requirements[21]. Stonehenge uses a disc scheduler to guarantee bandwidth between the storage server and the client.

[8] Looked at the integration of storage QOS and network QOS. Other solutions mentioned above looked at storage QOS and network QOS separately. [1] Proposed a priority based greedy algorithm for allocating storage server link network bandwidth to clients. Formulated mathematical models to calculate the required bandwidth. Solution implemented on object based storage system. Object based storage does not use file system instead it uses object attribute mechanism. The authors of [1] implemented a solution to calculate the needed network bandwidth for clients based on their SLO.Then in [1] they designed a priority based greedy bandwidth allocation to allocate the link network bandwidth. [8].

[22] Implemented a dynamic mechanism for providing resources on demand. The system uses Q-learning multi agent for managing each client access to the cloud resources. The authors of [22] Use throughput and CPU usage to measure bandwidth usage. Results shows reduction in idle bandwidth allowing low priority clients to use bandwidth while there is idle capacity.

[23] Developed SLED which is able to throttle very bursty workloads responsible for performance degradation. SLED is decentralized and therefore can be used to manage large storage systems. SLED main aim is to ensure effectiveness of storage systems by directing resources to those flows that do not have. However this approach may cause poor performance in high priority classes. In addition SLED is implemented on an FC SAN. Authors of [24] developed pTrans a framework for reservation guarantees based on directed acyclic graphs. However pTrans was found not to give accurate estimates for resource demand and available resources during run time which is crucial for dynamic resource allocation.[7] Developed bQueue which is framework for providing reservations and limits on storage systems. However Bqueue uses a simple round robin scheduler which has an advantage of low overhead but as determined in literature simple round robin end up causing delay especially in environments where there are packets of varied sizes and priorities.[25] Developed pShift which is a framework for providing I/O reservations and limits.Pshift uses estimates to provide optimal token distribution however it was found to be less scalable.

Motivated by the above discussion this study integrates a scheduler shaper that achieves better bandwidth utilization and achieves lower latencies better than the conventional solutions available[26].A NUM mathematical model for the optimal utilization of network bandwidth is formulated. The NUM mathematical model is solved using the Lagrange multiplier to find the optimal allocation value for each class of user. The study demonstrate through simulation that the proposed model is efficient in the utilization bandwidth and reducing latency. The proposed solution is implemented on a router positioned between the initiator and the target where the algorithm runs to avoid multiple copies of the same algorithm running in the network. This is expected to reduce overhead of processing multiple copies of the algorithm and eventually increase network performance.

3.DYNAMICBANDWDITH MANAGEMENT

3.1 Hierarchical Token Bucket Algorithm

In this section a description is made of the main features of hierarchical token bucket (HTB) specifically the implementation available in Linux traffic control[27]. HTB falls into the category of class based queuing disciplines[31][37]. A queuing discipline is a mechanism for queuing and dequeueing packets under the influence of an algorithm[29].HTB operates between the IP layer and the mac layer. In HTB flows are structured in a hierarchy of classes namely root, inner and leaf classes. All traffic goes through the root classes which is situated at the top. Inner classes are below the root classes with child classes as leaf classes. The leaf classes have no child classes however they have parent classes. Figure 1 illustrates the functioning of HTB.Flows control in each class is achieved by an internal token bucket which is populated with tokens limited by the rate a particular class is permitted to transmit[32].When a packet is transmitted belonging to a particular class its bucket subtracted with the number equal to the rate[15].

Each class configured with two rates that is rate bucket with tokens and a ceil bucket which contains ctokens (ceil tokens)[33].Tokens and ctokens is a measure of the amount of time a class occupies the scheduler output line. During transmission a class could either be in green, yellow or red states[34][36]. In the green state the class has sent less data than its allocated rate and therefore it can send more[35][33].In the yellow state the class has exceeded its guarantees rate but not ceil rate. In the red rate the class has sent more the ceil and cannot send any data.HTB uses DRR for sheduling.The class deficit is decremented based on the size of the packet.

Ctokens decrease by a ratio equal to $\frac{packet\ \ lengt\ h}{rate}$. This is the amount time a packet is in the scheduler queue. Ctokesn is added to the time elapsed after transmission .To take into account the time that elapsed since the last transmission in the same queue ctokens[12].To explain this further we use an example. Let t_2 be the current time and $t_1$be the last time since the last transmission.

$$\text{ctokens}(t_2) = ctokens(t_2) + (t_2 - t_1) - \frac{packect\ \ lengt\ h}{rate} \quad (1)$$

Given that *C i*s the capacity of the network in bps, any rate assigned to class *r*<*C*.Therefore $\frac{packet\ \ lengt\ h}{rate} > \frac{packet\ \ lengt\ h}{C}$. (2)

Equation 1 shows that when there is consecutive transmission from the same class the tokens constantly decrease [33][28].This is because the transmission is done at rate *r* therefore the value of *t2-t1* is added to the *C* pool which is equal to *packetlenght /r* which is less than *packetlenght/rate*[12].

If the expiration of the deficient for the current green class expires the scheduler might switch to the next green class. This is the case due to the working of DRR algorithm which is used in HTB as a scheduling algorithm[30].Scheduling algorithms are algorithms that determine the order in which packets are processed[28]. The DRR scheduling algorithm decrements the deficit after every transmission and in some cases it becomes zero or negative. In the mentioned cases *1/10* of rate is added to the deficit by default and then the scheduler can switch to the next green class if any. If there are no green classes the current one will continue to send until it is red or other become green[15]

Another case is when there is a bucket underflow[37].Bucket underflow is when ctokens bucket becomes empty which is an indication to the scheduler that the class is exceeding its ceil and therefore should switch to the next class.Ctokens takes the values in the interval [-cburst, cburst] where cburst is the peak rate[34].

When cburst is negative an underflow happens and the class status becomes red.On the other hand if ctokens goes above cburst the excess ctokesn are discarded. Since underflow has got a high priority, thedeficit expiration occursand the class stops sending data without putting into consideration the deficit. However if the deficit expires and other classes are red, the current transmitting class continues to send by adding a quantum value to deficit. It is important to configure a high cburst to ensure all the classes are green so as to allow transmission of all bytes from the current class before switching to the next one[12]. When a class has reached its ceil rate it queues packets until new tokens are available in a process known as policing. The working of the HTB is summarized in Figure1.

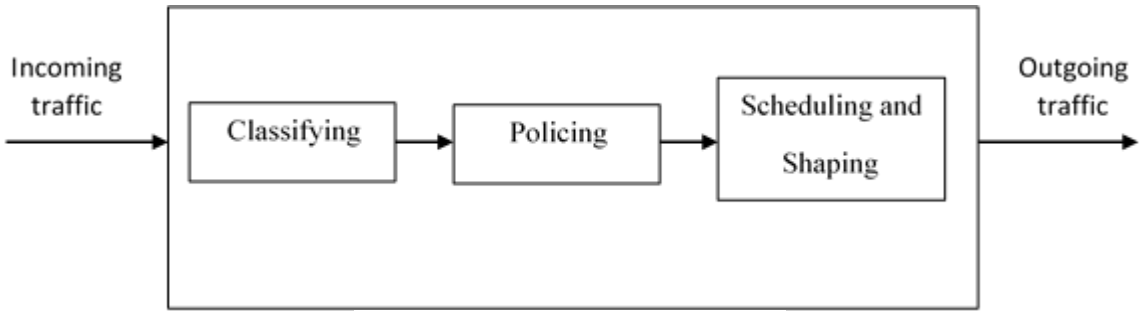

Figure1: Functioning of HTB

The key strength of HTB is bandwidth borrowing which ensures maximum utilization of the available bandwidth. Configurations for bandwidth borrowing is based on priority, high priority classes can borrow more bandwidth[32].

In HTB each class is configured with allowed rate(*R*),burst rate(*BR*) ,Guaranteed rate(*GR*) and rate that the class can borrow(*BW*).Therefore for any class *i*,in HTB we can define its allowed rate(*R*) as follows[15].

$$R_i = \min\left(BR_{i,} GR_i + BW_i\right) \quad (3)$$

Each class is configured with priority p and a quantum Leaf classes borrow bandwidth from their parents. If a leaf class has no parent then BW=0.For any class I with parent p and quantum I and priority p then the following equation holds[28].

$$BW_i = \left\{ \frac{Q_i R_p}{\sum_{j \in D} Q_{i\ where\ p_{j=p_i}}} \ if \min j \in D(p)\, p_i \ \geq p_i \right\} \quad (4)$$

From Equation 4 it is clear that rate is borrowed from parent and decided among all descendants levels based on priority according to quantum Q_i[15]

HTB cannot alone provide fairness and utilization, since it relies on prediction of output capacity of a link. We therefore need to include the current network statistics. Commercial routers do not provide optimization of bandwidth sharing for QOS by dynamically assigning bandwidth based on priority and network conditions [33].therefore this study proposes the traffic aware HTB for QOS provisioning based on priority[15].The proposed solution has been analyzed with a series of systematic experiments. The experiments have verified that the proposed HTB offers optimized bandwidth utilization and low latencies.

3.2 Limitation of Hierarchical Token Bucket Algorithm

A major component of providing QOS in a network is the scheduler. Packet schedulers are necessary in providing or ensuring bounded delay guaranteed bit rate and fair service allocation to all flows[38]. This can be achieved by solving the contention problem of a given resource and deciding on the sequence in which packets are transmitted from the node[31].

The router requires scheduling mechanisms to output packets arriving and ensure differentiated QOS[34]. The selection of an appropriate scheduling algorithm is key to providing QOS. A good scheduling mechanism should avoid unfairness between

packets[39]. Low priority packets should not be starved. In addition a good scheduler should provide good utilization constantly adjust the laws of their operation based on network statistics[28].

Packet schedulers are classified as either time stamped or frame based. Time stamped include the weighted fair queueing,worstcase fair queuing, virtual lock and self-clocked fair queueing.The advantage of time stamped scheduling algorithm is that they provide tight latency bounds and provide good fairness. However they have high complexity[34].

Frame based schedulers operate by rounds. Where each flow is served in a given round. Weighted round robin, deficit round robin and elastic round robin are frame based schedulers. These schedulers are easy to implement, however they have high latencies. This study considers specifically at DRR which is implemented in HTB.

DRR services flows in a round robin and succeeds in eliminating the unfairness of pure packet based round robin. However DRR latency become high when we have two flows with higher rate than the other. A good scheduling algorithm should have low computation cost, easy to implement, efficient and good fairness. DRR has a computation cost of O (1) though it does not have optimal fairness. This is because a flow continuously sends packets up to an amount of its deficit quantum which increases delay for smaller packets. Based on the deficiency of the DRR this study has put forward Hierarchical Priority Dynamic Deficit Round Robin scheduling algorithm (HPDDRR) technique that integrates traffic shaping and scheduling. HPDDRR uses a dynamic deficit counter that is generated based on the current network statistics for a given round. By using a quantum for the highest rate priority queue ensures high priority traffic is given preference hence achieving reduced delays. The hierarchy further ensures that flows are grouped based on classes which prevents interference.

4. BURST HANDLING

Storage I/O workloads are bursty in nature due to the device and application statistics and the location from where the device is being accessed from. These nature of I/O workloads burstiness makes it a challenge to achieve low latency as well as proportionate bandwidth allocation. In IP networks traffic shaping is the technique used to handle traffic by delaying low priority traffic in favor of high priority traffic. To implement traffic shaping two architectures are used that id direct exact sorting and rate based grouping. Direct exact sorting operates on per virtual connection queue at the input port. After the per virtual connection queue there are timing queues which are formed considering the incoming flows departure time. At the output port there aredeparture queue (DT) which sorts packets that conform or do not conform. Let the minimum rate of a connection be noted by X_i and the maximum rate be donated by X_i^*.Then the rate of a flow takes the range [X_i , X_i^*].For high speed connections let $\frac{1}{X_i^*}$ be the timing queue granularity. Let m be the number of timing queues, this follows that

$m \geq \frac{X_i^*}{X_i}$.The short comings of this technique is that m increases linearly if in the network we have flows with wide range rates. This makes the complexity of the architecture to be $O(\frac{X_i^*}{X_i})$.Therefore the direct exact sorting is not suitable for large networks with very wide range of rates.

With the rate based grouping architectures flows with similar rate are grouped together to reduce the range of rates. This means that each group can choose its own granularity. High granularity introduce jitter for high speed networks. Again the FIFO service in existing architectures does not control the interference between competing connections when multiple conforming cells await service. We need to handle large number of connections with wide range of bandwidth parameters. Handling large number of connections rate requires a large number of sorting queues.

To explain this further consider a shaping mechanism with flows varying from X_i to X_i^*,where $m>1$ being the rate differences factor between connections. As a result we have $n = Log_m\{\frac{X_i^*}{X_i}\}$ groups. For example give that $m=16$, $X_i = 1Kilobit/\text{Sec}$ and $X_i^* = 1gigabit/\text{Sec}$ this means that the number of groups will be $Log_{16}\{\frac{2^{30}}{2^{10}}\} = 5Groups$.A large bin granularity can introduce significant shaping delay and jitter to high rate connections.

To solve this connections can be grouped based on their bandwidth requirements allowing each sorting unit to select a different grain. With hierarchical architecture the shaper can select same sorting granularity for high rate connections to reduce delay.

5. METHODS AND MATERIALS

5.1 Materials

The implemented test bed includes five nodes three initiators, a target and a router. The router machine is equipped with two Ethernet ports. The three initiators are Virtual Machines each running a windows server 2016, with 4GB Ram and 26 GB target capacity. The target runs windows server 2016 with an 8GB RAM and 500 GB disk capacity. The router run Ubuntu 20.04 with a 4 GB RAM and a 500GB disk capacity.Parkdale disk benchmarking tool was used to simulate the reads and writes. In all the experiments a File size of 50MB was used unless otherwise stated.

5.2 Methodology

In achieving bandwidth management and traffic shaping the study adopted an experimental research design. Experiment is a research instrument that involves finding causal relationships between variables through the effect of manipulating one variable on another[42]. It is suitable for phenomenon with known variables or initial hypothesis that aimed at testing or manipulating a theory [41].It is also used to test and answer ‘how’ and ‘why’ research questions and lies in the deductive approach and positivism philosophy domain.

Experiments were set up to evaluate the proposed system on bandwidth allocation, bandwidth borrowing and burst handling. The proposed optimization of bandwidth management and traffic shaping was evaluated using the throughput and latency QOS metrics.

5.3 Model Formulation

Let *I* be a set of users for whom we want to allocate bandwidth to. A definition three QOS attributes that comprise the SLO for each class of user is made. These attributes are IO size, IOPs and response time.

The meaning of these attributes are as follows.

1. IOPs-I/O commands per second
2. Response time-time it takes for a request to receive a response.Constitues total latency
3. IO size-the amount of data read/written at a given instance.

S_i is defined to denote the SLO associated with a particular class of users where $i<=1<=n$

S_i={IOsize,IOPs, Response time} (4)

Let rsz_i denote the I/O request size of class i,IOP_irepresent IOPs for class i and rt_irepresent response time for class i.

Table 1.Estimated Storage Level Objective per user

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	1.6-3.2 ms
Power user	25 IOPS	100kb/s	1600kb/s	25000kb/s	1.3 ms

Let $B_R{}^i$be the total request bandwidth by class i based on the SLO.

The QOS attribute rsz_i,$IOP_i$and rt_i have got the following relationship

$$B_R^i = IOP_i * rsz_i \quad (5)$$

$$IOP_i = \frac{queudept\ h}{rt_i} \quad (6)$$

Therefore total bandwidth required by all the classes can be described as

$$B_{TR} = \sum_{i=1}^{n}(IOPi * rszi)\ \forall i \in I \quad (7)$$

Let B_{RW}^i represent the amount of bandwidth that configured for the class i to borrow. The total bandwidth to be borrowed B_{RW}^T can be described as

$$B_{RW}^T = \sum_{i=1}^{n} B_{RW\ \forall\ i \in I}^i \quad (8)$$

We describe the total bandwidth capacity of the network as B_C as

$$B_C = B_{TR} + B_{RW}^T \quad (9)$$

Let x_i the rate assigned to class i.Then the utility rate of class i is can be expressed as $\bigcup_i (x_i)$ which is a concave differentiable function. This means that if we increase an allocation to a given class it increases the total bandwidth allocate but it has no effect to the one class that has more resources already. This characteristic makes the utility function to be logarithmic in nature.

The study assumes that the network has a fixed capacity and therefore the goal is to maximize the collective utility o users in the network subject to network capacity constraints.

Therefore from the above narrative a maximization problem is formulated as follows

$$max \sum_i^n U_i(x_i) \quad (10)$$

Subject to $\sum_i^n x_i \leq B_{RW}^T$ (11)

$$x_i \geq 0, \forall i \in I \quad (12)$$

In the above equations $\bigcup_i (x_i)$ is the utility function of class i at ratex_i.I is the set of classes of users in the network. User i is identified with flow ratex_i.B_{RW}^T is the total excess bandwidth available. The study seeks to maximize to maximize the concave objective subject to linear constraints.

Based on proportional fairness the utility function we have

$$\bigcup_i (x_i) = \text{Log}\, x_i \quad (13)$$

Let P be a set of priority that is P= $\{p_i, i \in I\}$.

By introducing priority p_i we have

$$\bigcup_i (x_i) = p_i \text{Log}\, x_i \quad (14)$$

Let x_i^* be the optimal rate and x_i be the minimal rate.

Then for any allocation vector $x_i = \begin{Bmatrix} x_1 \\ x_2 \\ x_n \end{Bmatrix}$ we have an allocation equation as follows

$$\sum_i \frac{x_i - x_i^*}{x_i^*} \leq 0 \quad (15)$$

From equation 12 we note that for any allocation the sum of changes in the utilities will be less than zero[43]. That is if the rate of a given class i increases there is some rate of another class of users that decreases. The sum of this increases and decreases totals to a negative value[44].

If we assign the excess bandwidth based on priority proportional fairness the corresponding inequality is as follows

$$\sum_i p_i \frac{x_i - x_i^*}{x_i^*} \leq 0 \quad (16)$$

The α(alpha) fairness is used to investigate the different fairness criteria of max-min, minimum delay fairness and proportional fairness. The parameter α takes values in the interval (0, ∞)[45].

We define the α fair utility function as

$\bigcup_i (x_i) = \frac{p_i x_i^{1-\alpha}}{1-\alpha}$ Where α≥0,α≠1 (17)

Different values of α_i yield different fairness criteria

Case one of fairness we have α→1[46].

In this case maximizing the sum of

$\frac{x_i^{1-\alpha}-1}{1-\alpha}$ Provides the optimum value. (18)

The utility function for this case is

$$\bigcup_i (x_i) = p_i \text{Log}\, x_i \quad (19)$$

Case two of α fairness we have delay fairness where α=2[46].

Therefore the utility function for our case is:-

$$\bigcup_i (x_i) = \frac{p_i}{x_i} \quad (20)$$

If a class *i* is trying to transmit a file of size *rszi* and the rate allocated to this class is x_i,then the result is $\frac{rsz_i}{x_i}$ as the time taken to transfer the file. Case three is that of α fairness is that of minimum maximum fairness where α→∞[47].From the three cases of α fairness discussed above we can summarize the α fairness as follows

$$\bigcup_i (x_i) = \begin{cases} p_i \frac{x_i^{1-\alpha}}{1-\alpha}, & \alpha > 0, \alpha \neq 1 \\ p_i \text{Log}\, x_i, & \alpha = 1 \end{cases} \quad (21)$$

Equation 10 represents the priority proportional fairness. From this the study models a solution for priority based fairness utility maximization as follows

$$\text{Max } p_i \log x_i + p_1 \log x_1 + p_2 \log x_2 + p_3 \log x_3 \quad (22)$$

Subject to

$$x_1 + x_2 + x_3 \leq B_{RW}^T, x_1, x_2, x_3 \geq 0 \quad (23)$$

In order to solve the optimization problem, it is necessary to find the optimal allocationsx_1^*,x_2^*, x_3^*.To get the optimal allocations Langrage Multiplier on equation 8 is formulated. The variables x_1, x_2, x_3 are strictly positive. Again since the theory of convex optimization holds if the complementary slackness is satisfied, this means the Langrage multiplies to be used has to be positive. The Langrangian multiplier in this case is

$$L(x, \lambda) = p_1 \log x_1 + p_2 \log x_2 + p_3 \log x_3 + \boldsymbol{\lambda}(B_{RW}^T - x_1 - x_2 - x_3) \quad (24)$$

$$\frac{dL}{dx_i}, x_1 = \frac{p_1}{\lambda}, x_2 = \frac{p_2}{\lambda}, \text{ and } x_3 = \frac{p_3}{\lambda} \quad (25)$$

Using the constraint

$$x_1 + x_2 + x_3 \leq B_{RW}^T \quad (26)$$

$$B_{RW}^T = \frac{p_1}{\lambda} + \frac{p_2}{\lambda} + \frac{p_3}{\lambda} \quad (27)$$

$$\lambda = \frac{p_1 + p_2 + p_3}{B_{RW}^T} \quad (28)$$

Therefore

$$x_1^* = \frac{p_1 B_{RW}^T}{p_1 + p_2 + p_3} \quad (29)$$

$$x_2^* = \frac{p_2 B_{RW}^T}{p_1 + p_2 + p_3} \quad (30)$$

$$x_3^* = \frac{p_3 B_{RW}^T}{p_1 + p_2 + p_3} \quad (31)$$

In general

$$x_i^* = \frac{p_i \sum_{i=1}^n B_{RW}^i}{\sum_i^n p_i} \quad (32)$$

6. PROPOSED ALGORITHM DESCRIPTION

In this section the proposed HPDDRR which is a scheduler shaper is described which is meant to improve on latency and bandwidth utilization for flows.HPDDRR is a two stage mechanism which employs a single level hierarchy to aggregate flows into classes with similar priority and packet size. The key idea that enables the HPDDRR to alleviate the latency problem of DRR is the grouping of flows into classes with similar priority and almost similar packet sizes. This is an important since DRR is optimal when it acts with flows with similar packet sizes. The grouping of flows is so as to balance packet size per flow which will solve the problem of delays caused by large packets to small packets. The proposed algorithm begins by calculating the hit ration for each class of flows which is used to determine the priority of the flows. The priority of the classes is established using the equation $p_{i=\frac{h_i}{N}}$. Where h_i is the hit count of class *i* and *N* is the total number of hits.

Use of hit ratio is meant to ensure optimal utilization of bandwidth since the flows are allocated bandwidth proportional to their priority which is derived from their need. This reduces the chances of idle bandwidth or under allocation. Classification is done based on priority with flows of the same priority being put in the same class. From the classification the flows proceed to the shaper where packets that do conform to rates allocated are forwarded to the scheduler while those that do not conform are queued as they await bandwidth to be available.

During shaping, a flow is accepted if and only if the flow capacity is less than the guaranteed rate plus borrowing rate. Each classes/flow can be in one of the following states. Can borrow-bandwidth is sufficient and the number of packets sent is less than rate. May borrow-there are no tokens but it can be borrowed from parent class and the number of packets sent is greater than rate and less than ciel.Can't borrow-no bandwidth is available for borrowing that is the capacity of packets sent are more than ceil. Packets are classified using the u32 filter putting them into corresponding leaf classes. Bandwidth allocation is done using the HTB algorithim.HTB starts from the bottom of the class tree to find the class in the can send state until the class of the can send sate is found. If there are many flows in the can send state the algorithm will select the high priority classes. Each class sends its own quantum bytes by the means of poling until it's in the may borrow state. When the leaf classes is in May-borrow state it will borrow tokens from its parent's class until it is in can't send state.

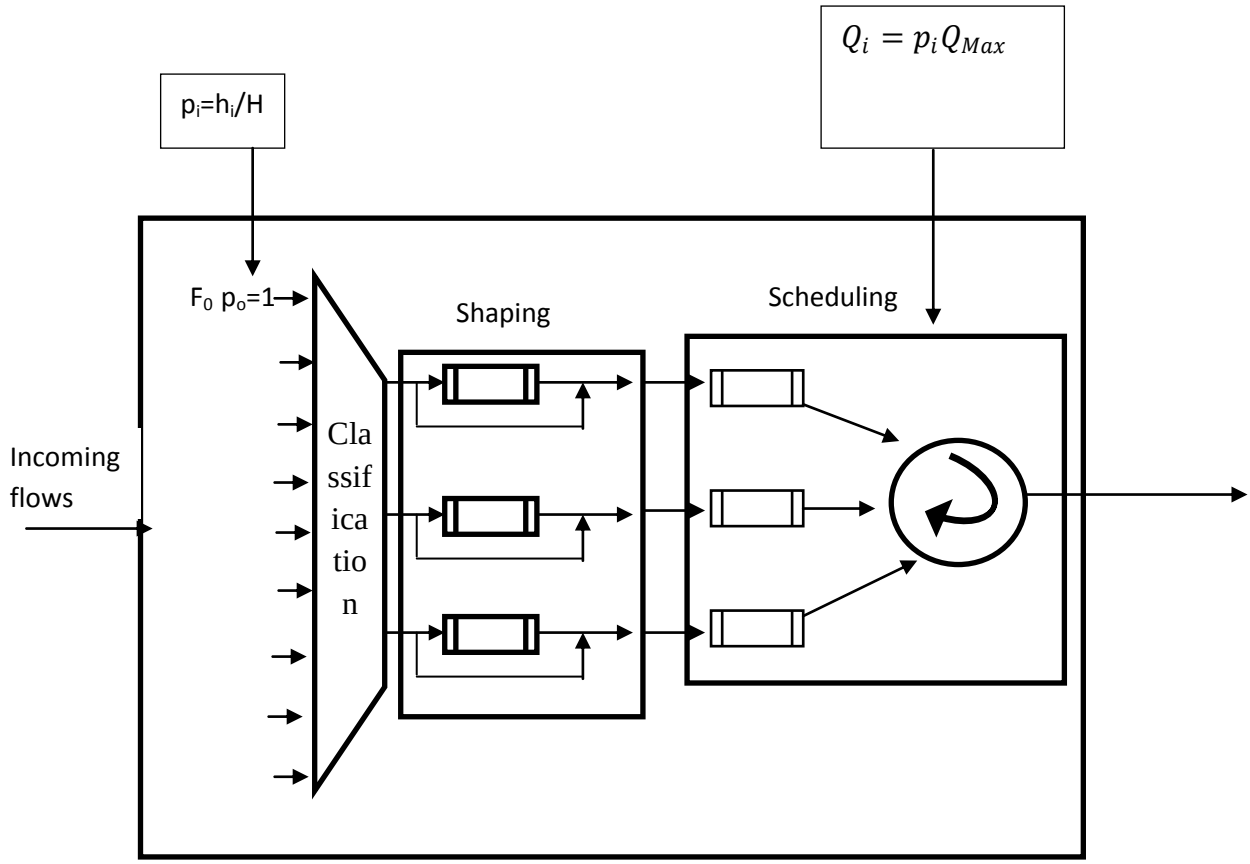

Figure 2: Architecture of the HPDDRR

To ensure the drop rates are low when bandwidth to be borrowed is not enough the lower priority classes releases some bandwidth at the same time ensuring that the users that releases the bandwidth their allocations do not fall below acceptable levels. Low priority classes are the ones that release bandwidth to ensure the high priority classes do not suffer from quality degradation. When there is enough free bandwidth available the proposed scheme gives the amount close to the maxima x_i^* otherwise if the available bandwidth is lower B_{RW}^T then the bandwidth allocation adjstments will be performed and the allocations for some low priority classes will be adjusted downwards and allocated the bandwidth of X_i.Flows are put into priority grades based on the SLO. When there is a need free the excess bandwidth the algorithm looks up at the low priority classes and checks the one that has bandwidth greater than the minima. If its finds that the current low priority class bandwidth is greater than the maxima the look up stops and the flow releases bandwidth to the high priority needy flow. If all the low priority flows cannot release enough bandwidth to satisfy the new flow, the high priority flows are queued.

A node with priority is assigned a bandwidth $\frac{p_i \sum_{i=1}^{n} B_{RW}^i}{\sum_i^n p_i}$where B_{RW}^i is the total available bandwidth. The higher the priority the more the bandwidth a flow receives.

At the scheduler the quantum for each round is calculated based on the rates of the highest priority class. Fig 2.illustrates the working of the proposed scheduler shaper.

Algorithim1: HPDDRR

INPUT: h_i, H,B_{TR}^i ,B_{RW}^i

OUTPUT:pkt_i,Q_i

Q_{max}: the biggest quantum size Possible. (Constant integer)

Q_i: Quantum the ideal rate a flow should receive in each round service (integer)

B_{TR}^i :Total available bandwidth

DC_i-Deficit from the previous round (integer)

P_i: Priority of class i

pkt_i: Packet belonging to queue *i*

B_{TR}^i : is total bandwidth allowed to class *i*

X_i^* ***:*** *is the maximum rate that a class can request*

Step1: Calculate the priority

$p_{i=\frac{h_i}{H}}$ //h_i total hits for class *i*,*H* is thetotal number of hits

Step2: Aggregate traffic into queues based on size and priority

//shaping

Step 3: Shape traffic

$B_{TR}^i = p_i \sum_{i=1}^{n} B_{RW}^i$

For each node

If $X_i^* \leq B_{TR}^i$ Then

Forward packets for scheduling

Else

Queue packets (delay packets)

//scheduling

Step4: Calculate the deficit counter based on priority

$DC_i=0;$

$Q_i = p_i Q_{Max}$

For each node do

While $Qi>0$ and queue i is not empty do

$Packet\ size=size\ (head\ (queue_i))$

If $packet\ size<=Q_i$then

Send (dequeue (queue i))

$NQi=Q_i$-$packet\ size$

If $packet\ size <=NQ_i$ then

Transmit packet and set $NQ_i=NQ_i$-$packet\ size$

Else

$DC_i=NQ_i$

$Queue_i$ ++

End if

End if

End for

Step 5: If(empty(*queuei*)) then

$DCi=0;$ repeat

The algorithm starts by shaping traffic. The maximum rate X_i^* is the maximum allowed rate for class i. B_{TR}^i is the total bandwidth allocated to class i.If the class rate is less than or equal to the available bandwidth the flows are forwarded to the scheduler otherwise they are delayed.

Next the packets arrive at the scheduler .In the scheduler there are n queues running from *1* to n that are served in a round robin fashion. Queue i belongs to class i.Deficit counter DC_istores bytes that a queue belonging to class i did not use in the previous round. At the beginning the DC_i is set to zero. Quantum Q_i represents the amount of capacity that each queue can use at each round of service. Each queue i belonging to class i has a different QOS requirement. For each queue i there is an associated priority. The requirements of flows belonging to a class i are established by calculating the priority i.The priority is used as the performance measure. Based on the priority which is dynamic, the quantum Q_i is calculated using formula

$Q_i = \frac{h_i}{H} Q_{Max}$ and allocated to each queue based on network statistics. $Q_{max}$is the maximum packet size that for any packet in an Ethernet network.

If the quantum size Q_i*>= to packet size*, then the packet is transmitted, else the algorithm moves to the next queue. Once a packet is transmitted its size in bytes is subtracted from the quantum Q_i to form NQ_i.If the NQ_i is not sufficient to transmit the packet in the head of the queue then the NQ_i is stored in DC_i to be used in the next round. Then the algorithm moves to the next round. In the end the total bandwidth receive by a queue i is the total quanta received by the queue. That is

$$B_{TR}^i = \sum_{i=1}^{n} Q_i \quad (33)$$

The difference between DDRR and DRR is that in DDRR the quantum is dynamic whereas in DRR the quantum is static.

7. EXPERIMENTAL SETUP

A real tested was implemented to show feasibility of the proposed approach. The evaluation was based on three service classes that is task users, knowledge users and super users. The implemented tested includes five nodes three initiators, target and a router. The router machine is equipped with two Ethernet ports. The three initiators are hosts each running a windows server 2016, with 4GB Ram and 26 GB target capacity. The proposed system was evaluated in terms of throughput and latency. Throughput was measured at the receiver's side. The bandwidth is allocated based on source destination IP addresses to assure a particular node generates traffic belonging to a particular class therefore having the same priority[48]. Whenever a node generates a flow, a request is sent to the router which includes required bandwidth and flow priority. Once the request reaches the router bandwidth management and burst handling is done based on the proposed algorithm. All experiments were run three times and the average value recorded.

8. RESULTS AND DISCUSSIONS

Experiments were performed to establish the performance of HPDDRR on bandwidth management and burst handling. Bandwidth management was implemented using the techniques of bandwidth allocation and bandwidth borrowing. For burst handling the technique used is traffic shaping.

6.1 Bandwidth Allocation

This experiment was performed to establish if HPDDRR is able to enforce proportional bandwidth allocation. An essential feature is that HPDDRR should allocate each class of users bandwidth proportional to their share in the range $[X_i$,$X_i^*]$.Three hosts running Parkdale and generating 64KB read/writes IO sizes were used. In addition DDRR was used for the host level scheduler. The proposed solution was run in the router with Hosts priority given allocations based on priority p_i set according to shares 1:2:3 for Hosts 1 to 3.Tabel 2 illustrates bandwidth utilization and latencies achieved when HPDDRR implements strict resource allocation. Figure 3 further depicts these results.

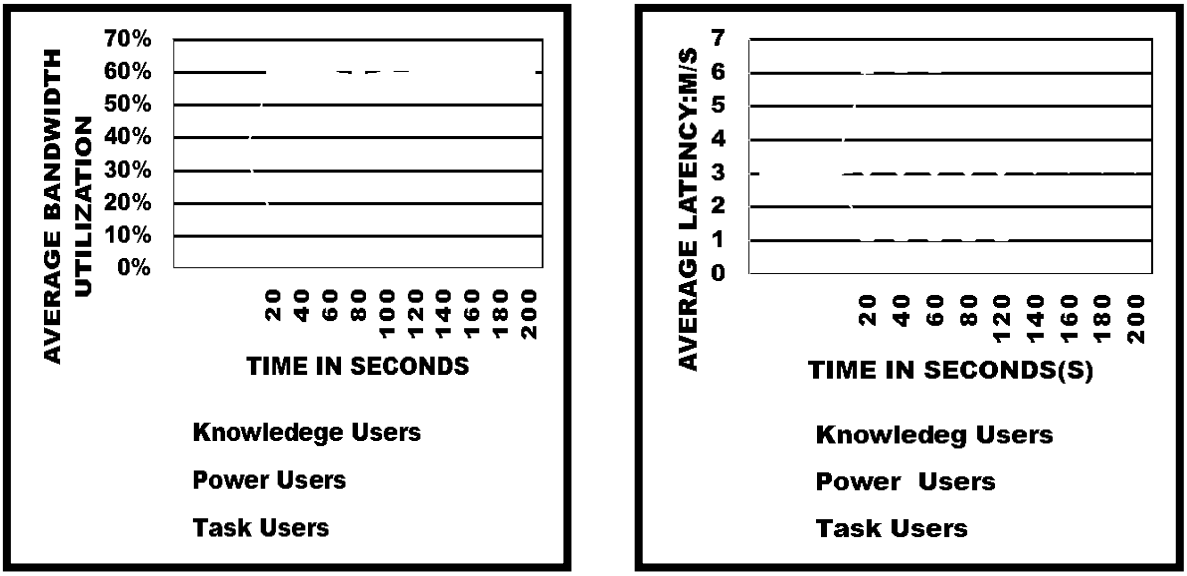

Figure 3 :(a)Bandwidth Utilization and (b) Latency received for three Classes of users with 1:2:3 share ratio.

From Figure 3 (a) it is observed that between time *t=0* to *t=20* all the classes of users seem hove equal utilization of bandwidth when HPDDRR is not activated. At *t=20* HPDDRR is activated and the results shows that it takes 10 seconds for the system converge to each class of users SLO.This convergence time is better than that of PARDA[49] and mClock[50] of 30 seconds. It with activation of HPDDRR that bandwidth utilized by each class of users is proportional to the overall *Pi* values in proportion to its shares. Power users received a percentage ratio of 60% throughput, Knowledge users received an average percentage ratio of 33% and task users attained an average percentage ratio of 16%, each matching their 3:2:1 ratio. From it is evidence that HPDDRR is able to maintain bandwidth allocation in proportion to the allocations based or their priority .Secondly it is observed that latencies achieved are consistent with the expected relationship between bandwidth allocation and latency. Higher bandwidth allocation results in smaller latency[5].

Figure 3 confirms the effectiveness of HPDDRR in bandwidth allocation where bandwidth is distributed based on priority. These results are similar to those obtained in Solutions like Stonehenge[49], Argon[51] and Aqua[52] support proportional allocation where users get a disk time share proportional to their weights.

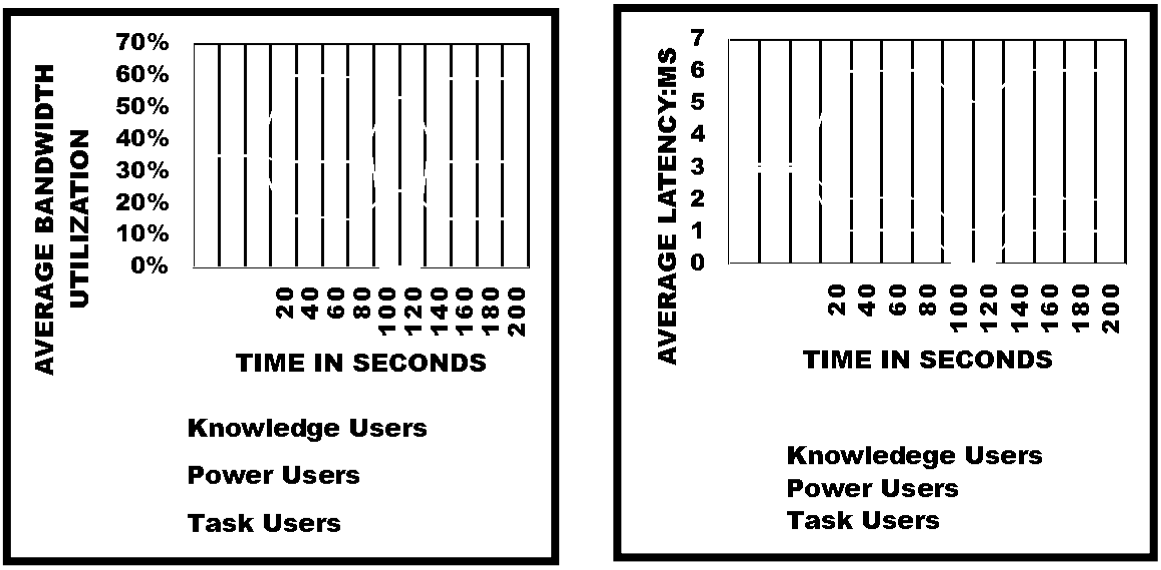

Figure 4: (a) Bandwidth utilization adaptation (b) Latency adaptation based on the share ratio 1:2:3

Table 2: Bandwidth Utilization and Latency observed when Strict Priority allocation is used.

Class of User	% Utilization	Average Latency(MS)
Task users	16	6.2
Knowledge users	33	3.1
Power users	60	1.2

6.2 Bandwidth Borrowing

Table 3: Bandwidth Utilization and Latency observed when Bandwidth Sharing is implemented.

Class of User	% Utilization)	Average Latency(MS)
Task users	17	5.8
Knowledge users	37	1.8
Power users	48	0.8

In this case the experiment intended to test the HPDDRR ability to implement bandwidth borrowing. It is expected that the proposed algorithm needs to be aware of changing bandwidth requirements and adopt accordingly based on priority. Experiments were carried out using a 1: 2: 3 share ratio. The three Hosts were used each generating a work load corresponding to the classes of task users, knowledge user and power users. Each host run a 64Kb random read/write IO size. All the Hosts are started at the same time with each hosts corresponding to power users stopped at between times *t=100* to *t=120* seconds. Table 3 illustrates an increase in average bandwidth utilization for task users and knowledge users as power users host was stopped at *t=100* seconds. The same is observed for latency as each class of user's experiences decreased latency.

Between times *t=0* and *t=20* HPDDRR is not activated and all users seem to utilize equal share of bandwidth as well as experience the same latencies. However at *t=20* HPDDRR is started. Figure 4 plots the bandwidthutilization and average latency observed by the three classes of users for a period of 200 seconds.

Note that in Figure 4, all flows get utilization proportional to their priority form *t=20* to *t=100*. Note that when the host for power users was stopped at *t=100* seconds, the now available capacity is distributed in a proportional manner. Note that the power users did not receive any extra share when restarted at *t=140* seconds since its arrival rate is the same to its SLO rate. These results are similar to those achieved in [53] where they were able to optimize throughput and latencies for consolidated hosts under SLO constraints. In addition there is a clear reduction in latency for knowledge users and task users when the power users host was stopped. This affirms the claim by[54] that when throughput increases latency reduces. The results in [53][7] demonstrated the same pattern where an increase in throughput caused a corresponding decrease in latency. Throughput optimization in [53] was also achieved through bandwidth borrowing so that when particular host is not using its share, the excess bandwidth is distributed to those Hosts that need it.This has been achieved by determining maximum bandwidth distribution based on demands. Optimization of bandwidth usage increases the throughput as it reduces the latency. Similar patterns were observed in results obtained in [7]

In conclusion of this section it is noted that latency can be reduced by managing bandwidth for each class of users, an observation supported by results obtained in[5][55].It also observed that with HPDDRR it takes 10 seconds to converge after power users host was stopped and then started . The convergence time is better than that obtained in PARDA[56] where the convergence time was 30 seconds. The results obtained in this section demonstrate the HPDDRR algorithm capability of supporting bandwidth borrowing as a feature of supporting bandwidth management in IP SANs.

6.3 Handling Bursts

Table 4: Throughput and Latency observed when Priority based Burst Handling is Implemented.

Class of User	Average Throughput(KB/s)	Average Latency(MS)
Task users	21223	6.1
Knowledge users	4500	3.4
Power users	25135	1.3

As mentioned earlier storage traffic is busty in nature due to application characteristics among other factors discussed in section 4 of this paper. This bursty nature of IO workload makes it difficult to implement proportionate bandwidth allocation as well as low latency. Experiments were run to establish how HPDDRR behaves when we have large bursts' busty arrival scenario is simulated when a class cases tries to send more that allowed burst value. A busty flow is most likely to miss deadline due to high delays. It is expected that the algorithm should be able to absorb bursts for other flows that send bursts equal or less than the allowed value. Solutions like PARDA[49] and mClock[50] use the idle credits to handle bursts. The flow with the greatest idle value is given the preference. Contrary to that, HPDDRR uses priority $P_i$to allocated idle bandwidthfor handling bursts. This ensures the high priority traffic always gets best of service

In the experiments three Hosts each running windows server 2016 configured with a 26GB data disk were used. Each host run a 1MB read/write workload. A 1MB IO size was used so as to generate more traffic compared to 64KB used in the previous experiments of this paper.

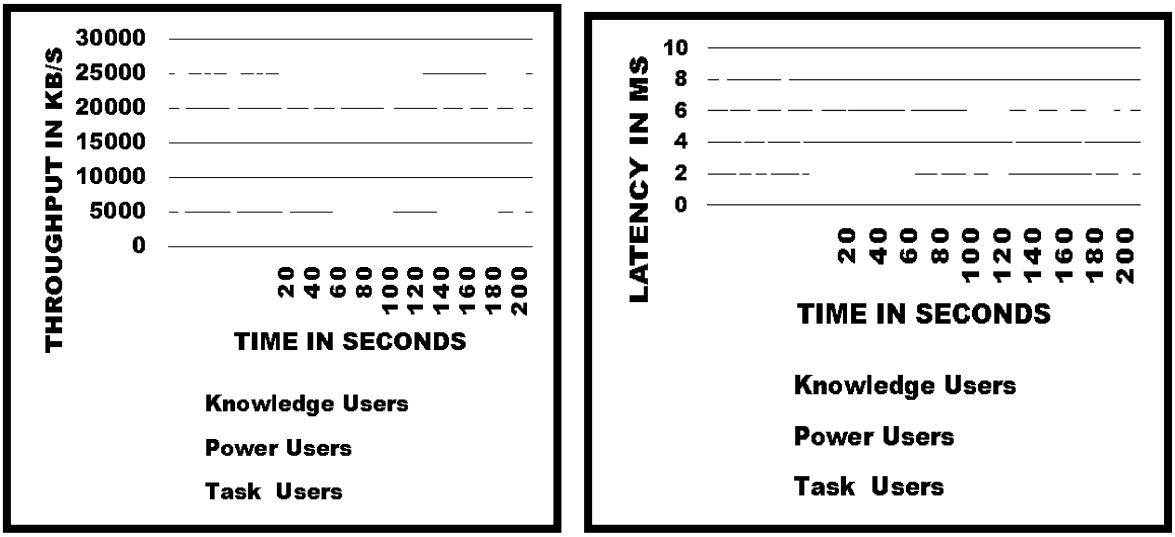

Figure5:(a)Knowledge users sends burst more than allowed values Power users and Task users send bursts equivalent to allowed value(b) Knowledge users don't meet the deadline whereas power and task users meet the deadlines.

To test how the system handles bursts, the following SLO parameters were used;

<IMB,25000KB/s,1.4MS>,<IMB,20000KB/s,2.4MS>,<IMB,5000KB/s,6.4MS>,for power users, knowledge users and super users respectively. Figure 5 plots the results for 200 seconds. Table 4 illustrates the bandwidth and latencies attained when doing burst handling.

From Figure5 demonstrates how the system behaves before HPDDRR is enabled and after it is enabled. It also demonstrated how the system adapts when HPDDRR is enabled to adhere to each class of users as specified in Table 1.From Figure 5 it is noted that for the first *20* seconds knowledge users send bursts of 1200 KB every *5* seconds. This is seen to reduce the throughput of task users and powers users as well as increase their latency. In this case all the class of users SLO is violated. At *t=20* HPDDRR is enabled and takes *10* seconds to converge to the SLO.This convergence time is better than that of PARDA[49] and mClock[54] of 30 seconds. At *t=60* the knowledge users send again send spikes of 1200KB/s each 5 seconds however this time other users are not affected. This can be attributed to the capability of HPDDRR to shape traffic. At *t=140* both power users and knowledge users send spikes of 2000KB every five seconds. However it is evident that the throughput for power user's increases but the latency does not unlike for knowledge users. This is due to the fact that HPDDRR uses priority to assign extra bandwidth for handling bursts unlike the knowledge users whose latency increase significantly due to lack of extra bandwidth for handling bursts which has been allocated to power users who have higher priority. This phenomena proves that HPDDRR uses priority to handled bursts. High priority flows will be given priority when it comes to allocation of spare bandwidth required for transmitting bursts of traffic.

From Figure 5 it is further evident that HPDDRR is able to absorb burst if the burst value is not higher that the burst size parameter and therefore able to handle burst for well behaving flows. These results are similar to those achieved in [54][25][24] where the authors were able to guarantee latencies based on SLO by shaping workloads. This was made possible by ensuring bursty and non-busty flows are smoothed in order to avoid head of line congestion.

9. CONCLUSION AND FUTURE WORK

In this paper the problem of bandwidth management and traffic shaping was studied. The paper proposes HPDDRR which uses hierarchical structure and a dynamic quantum to increase bandwidth utilization as well as reduce latencies experienced by flows in IP SANs.

Evaluation done on HPDDRR shows that it is able to provide proportional allocation of bandwidth to classes of users based on priority and adopt the utilization experienced by traffic classes of users based on network conditions .HPDDRR has also been proven through experiments that it is able to absorb bursts from classes of user's flows.

A hierarchical shaper can support more precise scheduling for the high rate traffic, this can significantly reduce cell shaping and jitter relative to existing approaches. With the hierarchical structure the sorting granularity for connection is reduced due to grouping. This reduces the implementation overhead and interference between competing connections.

As future work the research would like to test the performance of HPDDRR on non-storage systems.

10. ACKNOWLEDGMENTS

Special thanks goes to my supervisors Dr. Stephen Mutua and Dr. David Mwathi for their invaluable counsel during the entire research.

11. REFERENCES

[1] M. Karlsson, C. Karamanolis, and X. Zhu, "Triage: Performance Differentiation for Storage Systems Using Adaptive Control," ACM Trans. Storage, vol. 1, no. 4, pp. 457–480, 2005.

[2] X. Xuedong, "Research and Implementation of iSCSI-based SAN Static Data Encryption System," pp. 257–260, 2012.

[3] M. A. L. I. Imran, “Incast Mitigation in a Data Center Storage Cluster Through a Dynamic Fair-Share Buffer Policy,” IEEE Access, vol. 7, pp. 10718–10733, 2019.

[4] M. B. P. Martins and W. L. Zucch, “FCo oE an d iSC SI Per rforma ance Analys sis in T Tape irtualiz zation n Syste ems,” vol. 13, no. 7, pp. 2372–2378, 2015.

[5] Y. Cui et al., “TailCutter: Wisely cutting tail latency in cloud CDNs under cost constraints,” IEEE/ACM Trans. Netw., vol. 27, no. 4, pp. 1612–1628, 2019.

[6] V. Jaiman, S. Ben Mokhtar, V. Quéma, L. Y. Chen, and E. Rivière, “Héron: Taming tail latencies in key-value stores under heterogeneous workloads,” Proc. IEEE Symp. Reliab. Distrib. Syst., vol. 2019-Octob, pp. 191–200, 2019.

[7] Y. Peng and P. Varman, “BQueue: A coarse-grained bucket QoS scheduler,” Proc. - 18th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. CCGRID 2018, pp. 93–102, 2018.

[8] Y. Lu, D. H. C. Du, and T. Ruwart, “QoS provisioning framework for an OSD-based storage system,” Proc. - Twenty -second IEEE/Thirteenth NASA Goddard Conf. Mass Storage Syst. Technol., pp. 28–35, 2005.

[9] S. Sarmah and S. K. Sarma, “A Novel Approach to Prioritized Bandwidth Management in 802.11e WLAN,” 2019 IEEE 5th Int. Conf. Converg. Technol. I2CT 2019, pp. 1–5, 2019.

[10] J. L. Valenzuela, A. Monleon, I. San Esteban, M. Portoles, and O. Salient, “A hierarchical token bucket algorithm to enhance QoS in IEEE 802.11:Proposal, implementation and evaluation,” IEEE Veh. Technol. Conf., vol. 60, no. 4, pp. 2659–2662, 2004.

[11] D. Iswadi, R. Adriman, and R. Munadi, “Adaptive Switching PCQ-HTB Algorithms for Bandwidth Management in RouterOS,” Proc. Cybern. 2019 - 2019 IEEE Int. Conf. Cybern. Comput. Intell. Towar. a Smart Human-Centered Cyber World, pp. 61–65, 2019.

[12] Garroppo, Rosario Giuseppe, et al. "The wireless hierarchical token bucket: a channel aware scheduler for 802.11 networks." Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks. IEEE, 2005.

[13] Y. Wang, F. Xu, Z. Chen, Y. Sun, and H. Zhang, “An Application-Level QoS Control Method Based on Local Bandwidth Scheduling,” vol. 2018, pp. 1–10, 2018.

[14] Z. Zhou, Y. Yan, M. Berger, and S. Ruepp, “Analysis and Modeling of Asynchronous TrafficShaping in Time Sensitive Networks,” 2018 14th IEEE Int. Work. Fact. Commun. Syst., pp. 1–4, 2018.

[15] C. H. Lee and Y. T. Kim, “QoS-aware hierarchical token bucket (QHTB) queuing disciplines for QoS-guaranteed Diffserv provisioning with optimized bandwidth utilization and priority-based preemption,” Int. Conf. Inf. Netw., pp. 351–358, 2013.

[16] W. M. Zuberek and D. Strzeciwilk, “Modeling Traffic Shaping and Traffic Policing in Packet-Switched Networks,” vol. 6, no. 2, pp. 75–81, 2018.

[17] A. Elgabli, A. Elghariani, V. Aggarwal, and M. Bell, “QoE-Aware Resource Allocation for Small Cells,” 2018 IEEE Glob. Commun. Conf., pp. 1–6, 2018.

[18] B. Wu, B. Wu, H. Yin, A. Liu, C. Liu, and F. Xing, “Investigation and System Implementation of Flexible Bandwidth Switching for a Software-Defined Space Information Network,” IEEE Photonics J., vol. 9, no. 3, pp. 1–14, 2017.

[19] M. Song, “Minimizing Power Consumption in Video Servers by the Combined Use of Solid-State Disks and Multi-Speed Disks,” IEEE Access, vol. 6, pp. 25737–25746, 2018.

[20] H. Guo, “A Dynamic and Adaptive Bandwidth Management Scheme for QoS Support in Wireless Multimedia Networks,” vol. 00, no. c, 2005.

[21] P. Ramaswamy, “PROVISIONING TASK BASED SYMMETRIC QoS IN iSCSI SAN,” no. December, 2008.

[22] F. Uff, “A Lightweight Reinforcement-Learning-Based Multitenant Data Center,” pp. 331–336, 2020.

[23] D. D. Chambliss, G. A. Alvarez, P. Pandey, D. Jadav, and T. P. L. Ý, “Performance virtualization for large-scale storage systems,” 2003.

[24] Y. Peng and P. Varman, “PTrans: A Scalable Algorithm for Reservation Guarantees in Distributed Systems,” Annu. ACM Symp. Parallelism Algorithms Archit., pp. 441–452, 2020.

[25] Y. Peng, Q. Liu, and P. Varman, “Scalable QoS for Distributed Storage Clusters using Dynamic Token Allocation,” IEEE Symp. Mass Storage Syst. Technol., vol. 2019-May, pp. 14–27, 2019.

[26] E. Micha and N. Shah, “Proportionally fair clustering revisited,” Leibniz Int. Proc. Informatics, LIPIcs, vol. 168, 2020.

[27] B. Siregar, A. Fadli, and A. Hizriadi, “Controlling of Quality of Service in Campus Area Network Using OpenDaylight with Hierarchical Token Bucket Method,” 7th Int. Conf. ICT Smart Soc. AIoT Smart Soc. ICISS 2020 - Proceeding, pp. 8–12, 2020.

[28] D. Iswadi, R. Adriman, and R. Munadi, “Adaptive Switching PCQ-HTB Algorithms for Bandwidth Management in RouterOS,” Proc. Cybern. 2019 - 2019 IEEE Int. Conf. Cybern. Comput. Intell. Towar. a Smart Human-Centered Cyber World, pp. 61–65, 2019.

[29] K. Mathews, C. Kramer, and R. Gotzhein, “Token bucket based traffic shaping and monitoring for WLAN-based control systems,” IEEE Int. Symp. Pers. Indoor Mob. Radio Commun. PIMRC, vol. 2017-Octob, pp. 1–7, 2018.

[30] D. Iswadi, “Adaptive Switching PCQ-HTB Algorithms for Bandwidth Management in RouterOS,” pp. 61–65, 2019.

[31] Y. Qian et al., “A configurable rule based classful token bucket filter network request scheduler for the lustre file system,” Proc. Int. Conf. High Perform. Comput. Networking, Storage Anal. SC 2017, 2017.

[32] Sarmah, Satyajit, and Shikhar Kumar Sarma. "A novel approach to prioritized bandwidth management in 802.11 e WLAN." 2019 IEEE 5th International Conference for Convergence in Technology (I2CT). IEEE, 2019.

[33] S. Ren, Q. Feng, and W. Dou, “An end-to-end qos routing on software defined network based on hierarchical token bucket queuing discipline,” ACM Int. Conf. Proceeding Ser., vol. Part F1287, pp. 0–4, 2017.

[34] B. Siregar, A. Fadli, and A. Hizriadi, "Controlling of Quality of Service in Campus Area Network Using OpenDaylight with Hierarchical Token Bucket Method," 7th Int. Conf. ICT Smart Soc. AIoT Smart Soc. ICISS 2020 - Proceeding, pp. 1–5, 2020.

[35] W. Aljoby, X. Wang, T. Z. J. Fu, and R. T. B. Ma, "On SDN-enabled online and dynamic bandwidth allocation for stream analytics," arXiv, vol. 37, no. 8, pp. 1688–1702, 2018.

[36] Valenzuela, Jose Luis, et al. "A hierarchical token bucket algorithm to enhance QoS in IEEE 802.11: proposal, implementation and evaluation." IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004. Vol. 4. IEEE, 2004.

[37] S. Ren, "A Service Curve of Hierarchical Token Bucket Queue Discipline on Soft-Ware Defined Networks Based on Deterministic Network Calculus: An Analysis and Simulation," J. Adv. Comput. Networks, vol. 5, no. 1, pp. 8–12, 2017.

[38] A. N. Sanyoto, D. Perdana, and G. Bisono, "Performance Evaluation of Round Robin and Proportional Fair Scheduling Algorithm on 5G Milimeter Wave Network for Node Density Scenarios," Int. J. Simul. Syst. Sci. Technol., pp. 1–6, 2019.

[39] A. B. Mathews and G. Glandevadhas, "Improved Proportional Fair Algorithm for Transportation of 5G Signals in Internet of Medical Things," Int. J. Innov. Technol. Explor. Eng., vol. 9, no. 2, pp. 1810–1814, 2020.

[40] M. Saunders, P. Lewis, and A. Thornhill, Research for business students fifth edition,Pearson Education,2009. .

[41] Greener S. Business Research Methods.[e-book] Dr. Sue Greener and Ventus Publishing ApS. Available through:< http://www. bookbon. com>[Accessed 9 May 2011]. 2008.

[42] Winterton J. Business Research Methods ALAN BRYMAN and EMMA BELL. Oxford: Oxford University Press, 2007. xxxii+ 786 pp.£ 34.99 (pbk). ISBN 9780199284986. Management Learning. 2008 Nov;39(5):628-32.

[43] I. Guo, N. Langrené, G. Loeper, and W. Ning, "Robust utility maximization under model uncertainty via a penalization approach," Math. Financ. Econ., no. 2013, pp. 1–33, 2021.

[44] L. Vigneri, G. Paschos, and P. Mertikopoulos, "Large-Scale Network Utility Maximization: Countering Exponential Growth with Exponentiated Gradients," Proc. - IEEE INFOCOM, vol. 2019-April, pp. 1630–1638, 2019.

[45] Y. Wang, W. Wang, Y. Cui, K. G. Shin, and Z. Zhang, "Distributed Packet Forwarding and Caching Based on Stochastic Network Utility Maximization," IEEE/ACM Trans. Netw., vol. 26, no. 3, pp. 1264–1277, 2018.

[46] F. Zhang, R. Deng, and H. Liang, "An Optimal Real-Time Distributed Algorithm for Utility Maximization of Mobile Ad Hoc Cloud," IEEE Commun. Lett., vol. 22, no. 4, pp. 824–827, 2018.

[47] L. Gu et al., "Fairness-Aware Dynamic Rate Control and Flow Scheduling for Network Utility Maximization in Network Service Chain," IEEE J. Sel. Areas Commun., vol. 37, no. 5, pp. 1059–1071, 2019.

[48] L. Leonardi, L. Lo Bello, and S. Aglianò, "Priority-based bandwidth management in virtualized software-defined networks," Electron., vol. 9, no. 6, pp. 1–21, 2020.

[49] A. Gulati, G. Shanmuganathan, X. Zhang, and P. Varman, "Demand based hierarchical QoS using storage resource pools," Proc. 2012 USENIX Annu. Tech. Conf. USENIX ATC 2012, pp. 1–13, 2019.

[50] J, Lee E, Noh SH. I/O Schedulers for Proportionality and Stability on Flash-Based SSDs in Multi-Tenant Environments. IEEE Access. 2019 Dec 30;8:4451-65.

[51] Wachs, Matthew, Michael Abd-El-Malek, Eno Thereska, and Gregory R. Ganger. "Argon: Performance Insulation for Shared Storage Servers." In FAST, vol. 7, pp. 5-5. 2007.

[52] Wu, Joel C., and Scott A. Brandt. "The design and implementation of AQuA: an adaptive quality of service aware object-based storage device." In Proceedings of the 23rd IEEE/14th NASA Goddard Conference on Mass Storage Systems and Technologies, pp. 209-218. 2006.

[53] Li N, Jiang H, Feng D, Shi Z. Pslo: Enforcing the xth percentile latency and throughput slos for consolidated vm storage. InProceedings of the Eleventh European Conference on Computer Systems 2016,pp. 1-14.

[54] Y. Peng, "Latency Fairness Scheduling for Shared Storage Systems," 2019 IEEE Int. Conf. Networking, Archit. Storage, pp. 1–8.

[55] Wong, Theodore M., Richard A. Golding, Caixue Lin, and Ralph A. Becker-Szendy. "Zygaria: Storage performance as a managed resource." In 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), pp. 125-134. IEEE, 2006.

[56] Gulati, Ajay, Irfan Ahmad, and Carl A. Waldspurger. "PARDA: Proportional Allocation of Resources for Distributed Storage Access." FAST. Vol. 9. 2009.