FlowSM: Flow Scheduling Mechanism in Coordinated Mitigation of Large-Scale DDoS Attacks

Md Zillur Rahaman, Zhang Jian

Abstract— In this paper, we propose to construct mitigation of large scale of DDoS attacks based on "Weight Deficit" flow scheduling mechanism to improve the network maximize throughput and minimize the network congestion for mitigating DDoS attacks. Thus, the path of higher available bandwidth is mitigated of large-scale DDoS attacks. This flow scheduling mechanism able to maximize network throughput and to minimize the congestion with among elephant flows, mice flows, round robin flows. Flow Scheduling Mechanism (FlowSM) which is more adaptive to available bandwidth links and mitigate the congestion flows of the network and also mitigate the large scale of DDoS attacks. Weight deficit flow would to find the highest capacity of nodes in a network and assign weight ratio of those nodes. So it would able to provides a stable flow scheduling mechanism scheme which is adaptive to the available bandwidth links and mitigate large scale of DDoS attacks. Experiment results show that maximize network throughput of flow scheduling mechanism improves to mitigate the large scale of DDoS attacks. Simulation-based results shows that our technique effectively more defensive a victim server against various DDoS attacks. In this research paper, we propose a distribute approach to defend against DDoS attacks by coordinating flow scheduling mechanism across into the network.

Keywords— Data Center Network, Flow Scheduling , Distributed Denial of Service Attack, Network Security, Flow Control Mechanism

1. INTRODUCTION:

Network bandwidth is shared on dynamically between the random varying number of concurrently of this flow scheduling mechanism. We firstly discuss about the way of TCP realizes statistical bandwidth sharing, illustrating essential properties by means of packet level simulations. The high speed network usually deals with two main issues. The first is fast switching to get a good throughput. In this research paper, we present, the state-of-the-art switches are employing input queued architecture and get higher throughput. The second is providing QoS guarantees for wide range of applications. This is generally considered in output queued switches. For these two requirements, there have weight deficit flow scheduling mechanisms to support or get both better throughput and QoS guarantees in high speed switching networks for mitigating DDoS attacks.

In historically, bandwidth allocation networks has been at the mercy of TCP. TCP's model of allocation assumes that bandwidth should be shared to equally among contending flows. Many aim to minimize of per-packet flow latency or flow completion time (FCT) while others target bandwidth allocation while still others focus on sophisticated objectives like resource pooling, policy-based bandwidth allocation or co-flow scheduling.

• Corresponding Author. Tel.:+8801906834358;

E-mail: zillur.rahaman@yahoo.com

Dept. of Computer Science & Engineering

Packet scheduling refers to the decision process used to choose which packets should be served. It is the process of resolving contention for bandwidth. The target of a scheduling algorithm has to determine the allocation of bandwidth among the users and their transmission order. One of the most important tasks of a flow scheduling mechanism resides in satisfying the Quality of Service (QoS) requirements while efficiently utilizing the available bandwidth decide to send any given packet. A good choice that can help guarantee packet latencies through the router. The idea of the flow scheduling is to adapt the policy of transmission of packets in buffers according to the requirements of QoS for flows. Scheduling has a significant impact not only on the average delay but also on the buffer size. Scheduling is used to control the resources distribution between the classes of service.

Scheduling algorithm determines the allocation of the bandwidth among the users, flows or the services and classes. Priority scheduling can reduce the packet, delay, jitter and loss for the high priority traffic. The Strict Priority (SP) scheduling is simple and common solution. Weight Deficit Round Robin (WDRR) has well network scheduling discipline. Each packet flow or connection has its own packet flow in network interface controller. WDRR serves a number of packets for each non-empty flow and have their deficit weight. The number of packets served in proportion to the assigned weight and in inverse proportion to the size of the packets. Weight Deficit Round Robin is the most new variation of WRR that achieves better throughput approximation without knowing the mean packet size of each connection in advance.

DDoS is a type of DoS attack where the multiple compromised systems which are usually infected with a trojan virus in the single system of causing Denial of Service (DoS) attack. A very few flow scheduling mechanism of nodes while evaluating the performance of the security that had never done in before. It attempts to reduce flow congestion and improve the overall throughput of bandwidth allocation. The victims of such an attack can either be single node, set of nodes, the base station or even in the entire network. These attacks could further classified as flooding, ping of death, smurf attacks and flooding on victim's link. Basically, for DDoS attacks has a set of malicious entities towards a node or set of nodes. We use different way of DDoS attacks can be implemented. These are targeted flooding technique, movable node congestion, one owner and slave congestion.

On our main technical contribution is the transport design for solving the flow scheduling mechanism of congestion problem that converges significantly faster than prior work and is much more robust. The existing Weight Deficit Round Robin Algorithm couple these objectives and try to accomplish both simultaneously through weight variable and price variable at their links. These process is faster due to the need to balance between moving quickly towards the optimal allocation and avoiding or minimizing congestion or being under-utilization. FlowSM employs independent mechanism for the two goals by decomposing the task of solving network maximize throughput, minimize the network congestion, try to get higher bandwidth and it's reliable which will able to mitigate DDoS attacks. The rest of the paper is organized to introduce the deficit weighted based on connections flow scheduling mechanism architecture, describe elephant flow scheduling algorithm by the strict priority scheduling. We evaluate the performance of weighted based on connections flow scheduling algorithm in different traffic patterns for mitigating the large number of DDoS attacks.

2. Literature Reviews:

The authors is presented a survey of flow scheduling mechanism addressing the impact of various factors for DDoS attacks. There are some other cooperative techniques that are based on different flow scheduling mechanism in different purposes.

D. Y. Yau, J. C. S. Lui, F. Liang [2] The throttling is weight-fair because the traffics destination for the server are controlled by the leaky-buckets at the routers based on the number of users connected, directly or through other routers to each router. That was weight-fair technique for saving an internet server from DDoS attacks.

Li Chen, Kai Chen[3] Cloud applications generate a mix of flows with and without deadlines. Scheduling such mix-flows is a key challenge, trivially combining existing schemes for deadline / non deadline flows is problematic.

Mohammad Alizadeh, Abdul Kabbani [4] HULL leaves "bandwidth headroom" using Phantom Queues that deliver congestion signals before network links are fully utilized and queues form at switches for latency sensitive traffic to avoid buffering and the associated large delays.

Yau et al[6] viewed DDoS attacks as a resource management problem and proposed to distributed bottleneck resource as max-min fashion between level-k routers. K-Max-Min also similar to push-back has no mechanism to protect traffic of innocent hosts that share same path with the attacks. K-Max-Min also assume that all routers of level k accept to cooperate but it couldn't be a true assumption.

DefCOM[7] is another cooperative technique that makes an overlay between those routers that participate to detect and stop to DDoS attacks. DefCOM installed to classifier filters on nodes near traffic sources to distinguish attack packets from legitimate packets and ask other overlay node to rate limit traffic.

D-Ward technology[8] is used as classifier nodes in DefCOM. However, D-WARD is a very high expensive technique that burden large overheads to routers which we believe most routers reject to install firewalls such as D-WARD.

Sonia Laskara[11] Distributed Denial of Service attacks continue to instigate intense wars against popular e-commerce and content websites. QVMMA is an ideal faster real time solution to prevent DDoS attacks using Statistical Feature Vector Generation. Thus, it is concluded that QVMMA can based for effective DDoS prevention, mitigation in real time based on results generated Matlab simulation.

3. Releted Works:

We briefly discuss about related works that has informed and inspired our research design, algorithm, implementation, evaluation, analysis and especially work that did not discuss in before by using of FlowSM. For motivating by shortcoming of TCP, number of data center transport designs, network performance, their network utility maximization and low rate of DDoS attacks to differentiate of victim servers attack by QoS.

Flow Level Detection and Filtering of Low Rate DDOS: Flow level detection and filtering of low-rate DDoS. The recently proposed TCP targeted Low rate DDoS attacks send fewer packets to attack legitimate flows by exploiting the vulnerability in TCP's congestion control mechanism.

Mitigating DDoS Using Threshold Based Filtering Mechanism: Capability based approaches have been a major area of work since long time. They are robust against address spoofing attacks. However, they are vulnerable to a new type of attack called Denial-of-Capability attack. Also, bandwidth flooding is another serious issue. Dynamic threshold is for traffic monitoring, implemented over underlying basic capability approach in an effective attempt to mitigate vulnerabilities.

Scaling Flow Management for High-Performance Networks: Open Flow is a great concept but its original design imposes excessive overheads.

It can simplify network traffic management in enterprise and data center environments because it enables flow-level control over Ethernet switching and provides global visibility of the flows in the network. However, such fine-grained control, visibility comes with costs of the switch implementation, costs of involving the switch's control-plane, the distributed-system costs of involving to the Open Flow controller and frequently both on flow setups and especially for statistics-gathering.

Dynamic Flow Scheduling for Data Center Networks: Today's data centers offer tremendous aggregate bandwidth to clusters of tens of thousands of machines. However, because of limited port densities in even the highest end switches and data center topologies typically consist of multirooted trees with many equal-cost paths between any given pair of hosts. Existing IP multipathing protocols usually rely on per-flow static hashing and can cause substantial bandwidth losses due to long-term collisions. A scalable dynamic flow scheduling system that is adaptively schedules a multistage switching fabric to efficiently utilize aggregate network resources.

Finishing Flows quickly with Preemptive Scheduling: Today's data centers face extreme challenges in providing low latency. However, fair sharing a principle commonly adopted in current congestion control protocols is far from optimal for satisfying latency requirements. Preemptive Distributed Quick (PDQ) flow scheduling is a protocol designed to complete flows quickly and meet flow deadlines. PDQ enables flow preemption to approximate a range of scheduling disciplines.

Fast and Flexible Bandwidth Allocation in Data Centers: Flow Scheduling Mechanism a novel transport design that provides flexible and fast bandwidth allocation control. It enables operators to specify how bandwidth is allocated among contending flows to optimize for different service level objectives such as strict priority flows, weighted fairness, minimizing flow completion times, multipath resource pooling, prioritized bandwidth functions.

4. Designs & Algorithms:

We use this flow scheduling mechanism to clarify our design and develop to the main themes of research are here in below. We are using two different kinds of flow scheduling mechanism in switches for distributing data packet and observe to mitigate of large number of DDoS attacks.

Strict Priority Flow Scheduling: A port is idle and the packet with highest priority buffered at that port is dequeued sent out which is called Strict Priority (SP). When the network is ideal then the packet will send out by their priority based. However this priority does not see their weight and cost of their building networks. As the figure (01) say that if A packet come to queue and then it will send out by their priority based which is strictly coordinate with those packet priority. After that other packets will come on the next queue and do it same process. It is the normal process of the network which is used in last few years before. For that strict priority based flow scheduling mechanism is mainly used in efficient feedback compression and reduce the cost of base station synchronization and reduce feedback delay.

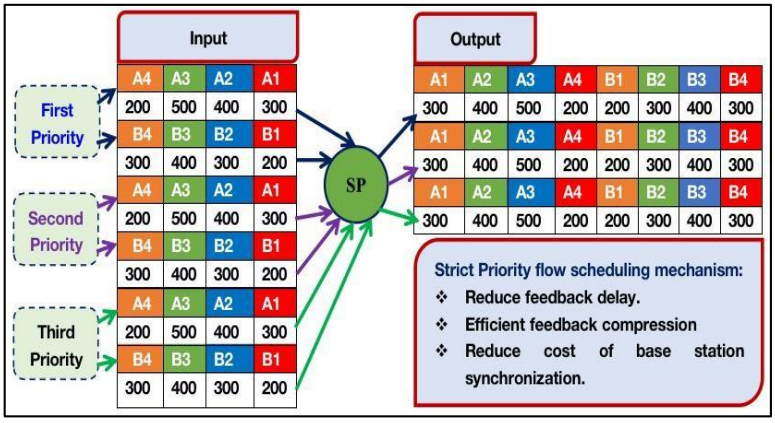


Figure (01) : Strict Priority Flow Scheduling

Relative strict-priority flow scheduling provides strict-priority flows within a shaped aggregate rate. For example, it allows you to provide 1 Mbps of aggregate bandwidth to subscriber with up to 500 Kbps of the bandwidth for low-latency traffic. If there is no strict-priority traffic, the low-latency traffic can be used up to the full aggregate rate of 1 Mbps. Relative strict priority differs from true strict priority which it can implement the aggregate shaping rate for both strict and non strict traffic with true strict priority. We can shape the non strict or the strict traffic separately but cannot shape the aggregate to a single rate. The best application of relative strict priority is on Ethernet where you can shape the aggregate for each VLAN to a specified rate and provision a strict and non strict queue for each VLAN above the shaped VLAN node.

To use relative strict priority, we can configure strict-priority queues above the VLAN scheduler node thereby providing for strict-priority scheduling of the queues within the VLAN. We also can configure relative strict priority without using QoS traffic-class groups which causes strict-priority queues to appear in the same scheduler hierarchy as the non strict queues. Relative strict priority provides low latency only if we under subscribe the port by shaping all VLan on the port so that the sum of the shaping rates is less than the port rate. The port will not become congested and the latency caused by the round-robin behavior of both the HRR and cell schedulers is nominal. In these under subscribed conditions, the latency of a strict-priority queue within each VLAN is calculated as if the VLAN drains onto a wire with bandwidth equal to the shaped rate.

Different Priority Assignment Schemes: The next, we compare three different schemes for assigning packet priorities with increasing degrees of complexity. For each packet transmitted the priority field is set to be: (i) number of bytes thus far sent from flows; (ii) flow size in bytes; (iii) remaining flow size in bytes. The first scheme is the simplest as it does not require knowledge of flow size. The second and third schemes both require flow size information but the second is simpler

since the priority number is decided once and remains constant for all packets flow. In switch implementation since we do not need starvation prevention mechanism.

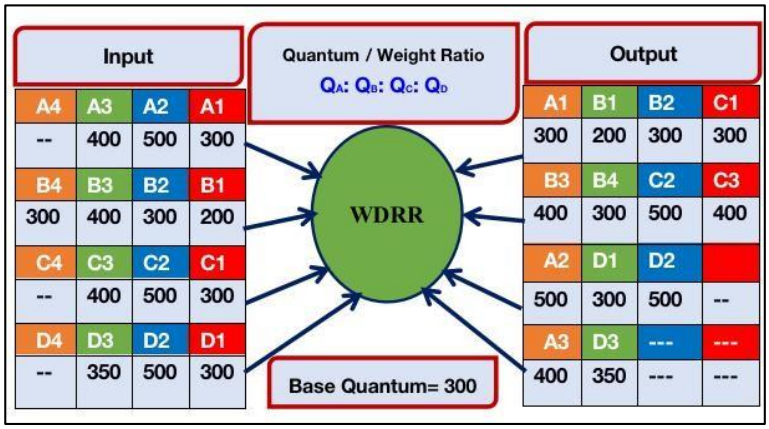


Figure (02) : Weight Deficit Round Robin Flow Scheduling (WDRR)

Weight Deficit Round Robin Flow Scheduling (WDRR): Weight Deficit Round Robin (WDRR)flow scheduling is increasingly popular in modern switches to determine the schedule packets when multiple input ports are trying to transmit out a single output port. Weighted fair queuing (WFQ) is the better standard of fairness in flow scheduling. We can define scheduling mechanism to be where two streams are able to achieve the same overall throughput on a network. WDRR actively monitors the average size of the frames in input queues and adjusts the weights of the queues in order to favor the queues with smaller frames in them. This ultimately overcomes identified fairness issue and allows streams of small frames to have the same throughput of larger frames. Using a well-known fairness index, we have plotted the fairness of the system before and after implementing our proposed algorithm. WDRR solves max-min weight flow scheduling algorithm problems and make it faster and more reliable than existing approach. WDRR is to decouple the underlying mechanisms for maximizing network utilization, minimizing the network congestion time with getting higher bandwidth and achieving the optimal relative rate allocation to mitigate the large number of DDoS attacks.

Switch Design: The majority of the functionality resides at the leaf switches and spine switches. The source leaf makes load balancing decisions based on per up-link congestion metrics, derived by taking the maximum of the local congestion at the up-link and the remote congestion for the path of the destination leaf that originate at the up-link. The remote metrics are obtained via feedback from the destination of leaf switch which opportunistically Piggyback values in the congestion from leaf switches. Load balancing decisions are made on the first packet of each flows. The sub-sequence packet use the same up-link as fully active. The source leaf uses to keep track of active Flow-lets and their up-link.

Packet Format Design: To leverages the VLAN encapsulation format used for the overlay to carry the following state.This field partially identifies the packet's path set by the source leaf switches port. The number of the up-link packet is sent on and is used by destination leaf to aggregate congestion metrics before they are feedback to the source.This field is used by switches along the packet's path to convey the extent of congestion.These two fields are used by destination leaves to piggyback congestion information back to the source leaves. When flows start at line rate. Practically, this is accomplished by using an initial window size equal to the bandwidth-delay product (BDP) of the link (32 packets in our simulations. We use SACKs for every packet acknowledgment, we do additive increase as in standard TCP. There are no fast transmits, dupACKs mechanism. Packets drops are only detected by timeouts whose value is fixed and small. If it fixed threshold number of consecutive time outs occur then it indicates a chronic congestion collapse event. In this case, the flow enters into probe mode where it periodically re-transmits minimum-sized packets with one byte payload and re-enters slow-start once it receives an acknowledgment.The only goal is that avoid excessive and persistent packet drop which is simple design accomplishes.

Flow Scheduling Rate Control Design: We do not need to worry about keeping queue occupies small to control queuing latency. Since packets are scheduled based on priority even if large queues do form in the mechanism. There would be no the latency for high-priority traffic. However, there is one corner case where a limited form of rate control is necessary. Specifically, whenever a packet traverses multiple hops only to be dropped at a downstream link some bandwidth is wasted on the upstream links that could have used to transmit other packets. This is especially problematic when the load is high and multiple elephant flows collide at the downstream.

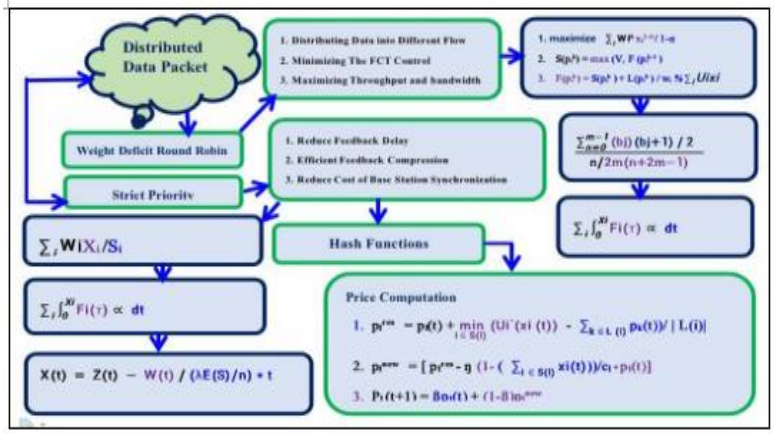


Figure (03) : Flow Scheduling Mechanism Flowchart (1)

Algorithms:

WDRR stands for Weight Deficit Round Robin that can be used for DDoS detection and mitigation in real time

	Name of Algorithm	Equation of algorithm
1.	Utility Function of Algorithm	maximize $\sum_i U_i(X)$; Subject to $Rx <= c$
2.	Fairness Algoarithm	$\sum_i W_i X_i^{1-\alpha} / 1-\alpha$
3.	Weight Deficit Round Robin	$\sum_i W_i^{\alpha} X_i^{1-\alpha} / 1-\alpha$ $S(p_i^k) = \max(V, F(p_i^{k-1}))$ $F(p_i^k) = S(p_i^k) + L(p_i^k) / w_i$
4.	Minimizing Flow Completion Time	$\sum_i W_i X_i / S_i$
5.	Variant of Load Balancing	$X(t) = Z(t) - W(t) / (\lambda E(S)/n) * t$
6.	Bandwidth Functions	$\sum_i \int_0^x F_i(\tau) \propto d\tau$
7.	Hash Function Implementation for Mitigating DDoS Attacks	$\sum_{n=0}^{m-1} (b)(b+1) / 2 / (n/2m(n+2m-1))$

Figure (04) : Flow Scheduling Mechanism Flowchart (2)

Fairness Algorithm: Various notions of fairness can be expressed simply by changing the shape of the utility functions.

$$\sum_i W_i X_i^{1-\alpha} / 1-\alpha$$

The α- fair class of utility functions represented in the first row of figure 6 enable an operator to express different preferences on the fairness / efficiency trade-off curve by varying α, a non-negative constant. α = 0 is purely utilitarian, maximize overall throughput without concern for fairness. As α increases, flow scheduling solution gets "more fair", eventually converging to the max-min fair allocation as α → ∞. An important case is α = 1, which is a compromise between these extremes and is called proportional fairness. α-fair utility functions can also be generalized to express relative priorities using different weight multipliers for different flows as shown in the figure 6.

Weight Deficit Round Robin Algorithm: WFM is a flow based algorithm that achieves asymptotically 100% throughput with no speed up while providing QoS.

$$\sum_i W_i^{\alpha} x_i^{1-\alpha} / 1-\alpha$$

$$S(p_i^k) = \max(V, F(p_i^{k-1}))$$

$$F(p_i^k) = S(p_i^k) + L(p_i^k) / w_i$$

For these reasons, in addition to the demand for high throughput on routers or switches with input queued architecture. There is an increasing need for supporting applications with diverse performance requirements where QoS is guaranteed. However there has been a restriction to provide QoS guarantees in an input queued switch input queued switch is scalable but lead to some packets not being promptly transmitted across switch fabric because en-queued packets can not be isolated which may lead to violating QoS. Therefore the goal of providing QoS guarantees in the input queued switch is to design a scheduling algorithm which can provide QoS requirements so that queued packets are transmitted across the switch scheduling mechanism promptly.

In this paper, we propose a scheduling algorithm for providing QoS guarantees and high throughput in an input queued switch for mitigating the large-scale of DDoS attacks. The proposed algorithm called Weight Deficit Round Robin (WDRR) which is a flow based algorithm that provides bandwidth allocation. The WDRR in input queued switches is unique in a sense that the selection right and corresponding matching mechanism based on virtual finishing time of WDRR is done at the output port where the number of connections to the output ports and the virtual finishing time stamps already computed and transferred by input ports are involved.

Minimizing Flow Completion Time Scheduling Algorithm: Size-based scheduling policies that are effective for minimizing (average) flow completion time can also be approximated within the data framework as shown in the figure 6.

$$\sum_i W_i X_i / S_i$$

The utility functions are linear in the rates and associate a weight to each flow inversely proportional to its size (si). This objective also performs very well in the multilink case. Similarly, the weights can be chosen inversely proportional to the remaining flow size or flow deadlines to approximate Shortest-Remaining Processing-Time (SRPT) or Earliest-Deadline-First (EDF) scheduling for meeting deadlines.

Using Swift Parameter to Detect the DDoS attacks : A distributed algorithm that calculates the deficit weights for the flows such that the weighted max-min rate allocation solves the Scheduling mechanism problem. Swift provides the abstraction of a network with guaranteed high utilization and weighted max-min allocation, where the flow weights can be set dynamically. This allows the relative bandwidth allocation of the flows to be controlled without having to worry about high utilization or network congestion. xWI leverages this capability to quickly search for the deficit weights and link prices which now solely act as a coordination signal not a measure of congestion enabling xWI to converge quickly and safely.

Load Balancing Hash Algorithm: The switch balances packet load across multiple links in a port channel by calculating a hash value based on packet header fields. The hash value determines the active member link through which the packet is transmitted.

$$X(t) = Z(t) - W(t) / (\lambda E(S)/n) * t$$

This uneven distribution is avoided by performing different hash calculations on each switch routing the paths. The port-channel load-balance command specifies the seed for hashing algorithms that balance the load across ports comprising a port channel. Available seed values vary by switch platform.

Variant of Load Balancing: Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource. Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy. Load balancing usually involves dedicated software or hardware such as a multilayer switch like leaf switches and spine switches and Servers.

Bandwidth Variation Equation: The available bandwidth of a network path is an important performance metric and its end-to-end estimation has recently received significant attention. The available bandwidth is the maximum throughput that the path can provide to an application and given the path's current cross traffic load. Measuring available bandwidth is not only for knowing the network status but also to provide information to network applications on how to control their outgoing traffic and fairly share the network bandwidth.

Hash Algorithm and Analyzing: A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. The values are used to index a fixed-size table called a hash table. Use of a hash function to index a hash table is called hashing or scatter storage addressing. Hash functions and their associated hash tables are used in data storage and retrieval applications to access data in a small and nearly constant time per retrieval and storage space only fractionally greater than the total space required for the data or records themselves. Hashing is a computationally and storage space efficient form of data access which avoids the non-linear access time of ordered and unordered lists and structured trees and the often exponential storage requirements of direct access of state spaces of large or variable-length keys. Use of hash functions relies on statistical properties of key and function interaction which it worst case behavior is intolerably bad with a vanishing small probability and average case behavior can be nearly optimal. Hash functions are related to checksums, check digits, fingerprints, lossy compression, randomization functions, error-correcting codes, and ciphers. Although the concepts overlap to some extent, each one has its own uses and requirements and is designed and optimized differently. A hash function takes as input a key which is associated with a record and used to identify it to the data storage and retrieval application. The keys may be fixed length like an integer or variable length, like a name. The output is a hash code used to index a hash table holding the data or records or pointers to them.

5. IMPLEMENTATIONS:

I want to implement a sophisticated flow scheduling mechanism to make bandwidth faster and more reliable and more flexible at end host by using of Weight Deficit Round Robin Flow Scheduling Algorithm provides more secure and to mitigate the large scale of DDoS attacks. A prototype implementation of flow scheduling mechanism with including the hardware like as switches (Leaf switches & Spine switches), Web Server / TCP server and software like Matlab Flow Scheduling Analysis, ns3 Network Simulator / Packet Tracer Network Simuator and end-host stack are beyond the scope of this research paper and is part of our future work. Here, we can briefly analyze the feasibility of its implementation.

Implementation of Utility Function: The utility function depends on the bandwidth allocation objective that the operator wishes to achieve. We pick five popular and broad bandwidth allocation policies and show how they can be expressed using utility functions and a similar exercise can be carried out for other policies.

Switch Implementation for Strict Priority: Priority flow scheduling mechanism and scheduling dropping are relatively simple to implement using

well known and widely used hardware primitives because in data centers has different types of data packet. If the packet is priority based then the data packet can be stored in client sever which is web server/ hosts. The web server provides the data packet with including priority(High priority, Medium Priority, Lowest Priority) based scheduling and connected two leaf switches. Leaf switches helps to distributed data packet into different flows.

Typologies of Servers: We simulate a data center network built using a leaf-spine architecture. There are a total of 4 (Client Web Servers & Destination Web Servers) servers connected to 2 leaf switches with 10 Gbps links. Each leaf switch is connected to 2 spine switches using 40 Gbps links, thus ensuring full bisection bandwidth. The switches are modeled as standard output-queued switches, with a buffer of size 1 MB per port. We chose this large limit to avoid complications for comparing the convergence times of different algorithms which are sensitive to packet drops. All of the implemented schemes target a small queue occupancy and thus avoid packet drops well below this buffer size. The queue occupies are typically only a few packets at equilibrium. The network RTT is 4 µs.

Large Scale of Simulations by NS2: We now compare a using of few priority queues in existing switches with flow scheduling mechanism. Our results confirm that while this mechanism provides good performance with a sufficient number of priority queues. It is still worse than flow scheduling mechanism and the performance is sensitive to the value of the thresholds used and also how the switch buffer is shared among the priority queues. We simulate the web search workload for three scenarios with 2, 4, and 8 priority queues per flows port. The queues at a port share a buffer pool of size 225KB (150 packets). We reserve 15KB (10 packets) of buffer per queue and the rest is shared dynamically on a first-come-first-serve basis. We observe that as expected the average overall FCT part (a) improves as we increase the number of priority queues and is close to flow scheduling performance with priority queues. We observed a similar trend in the average FCT across small, medium, and large flows. In figure 10 shows that there is a significant increase in the 99th percentile FCT for the small flows at high loads in the 8-queue case. This is because with 8 queues, 80 out of the total 150 packets are reserved and leaving only 70 packets to be shared among the queues. Thus at high load during some bursts, the high priority queue runs out of buffers and drops packets and increasing tail latency. This demonstrates the need for carefully tuning the buffer allocations for each priority queue for good performance.

Sensitivity to thresholds: Finally, we explore the sensitivity of the performance with a few priority queues to using the "right" thresholds for splitting traffic. In figure show 7 a comparison of the 4-queue system with optimal thresholds with a reasonable values that splits flows equally across the 4 queues which the smallest 25% of flows are assigned to the highest priority queue, second smallest 25% to the second highest priority etc. The plot shows the aver-age FCT across all flows. We find a fairly substantial improvement with the optimized thresholds. At 80% load, the average FCT is reduced by more than 30% with more substantial performance gaps for the tail latency for short flows. This confirms that the thresholds for splitting traffic across limited priority queues need to be chosen carefully. By allowing an essentially unlimited number of priorities, flow scheduling mechanism does not require any tuning and is not sensitive to parameters such as thresholds, minimum reserved buffer per priority queue and overall buffer size.

Switch implementation: Priority scheduling and dropping are relatively simple to implement using well known and widely used hardware primitives because flow scheduling mechanism of switches have very small buffers. This can be done in parallel for all 32 packets but it is preferable to do it sequentially on smaller blocks to reduce the required gates and power-draw. Assuming a 32 blocks compare that checks 32 flow-ids at a time we require at most 3 clock cycles for all 32 packets. Hence we need a total of clock cycles to figure out which packet to dequeue which is well within the budget of 40 clock cycles. The analysis for the enqueuing is simpler since the only operation there is the operation performed by the binary tree when the queue is full. A number of optimization can further simplify the FlowSM switch implementation. For instance, we could use a hash of the 5-tuple as the flow-id to reduce the width of the bit-wise flow-id comparators. A fairly short hash 8–12 bits should suffice since the total number of packets is small and occasional hash collisions only marginally impact the scheduling order. Moreover, if we restrictly provides flows by the priority assignments such that a flow's priority does not increase over time by using absolute flow size as the priority instead of remaining flow size. We would not need the starvation prevention mechanism and could get rid of the flow-id matching logic completely. Our results indicate that using absolute flow size is almost as good as remaining flow size for realistic flow size distributions found in practice. Note that our switches do not keep any other state nor are they expected to provide feedback nor do they perform rate computations. Further, the significantly smaller buffering requirement lowers the overall switch design complexity and to die the area.

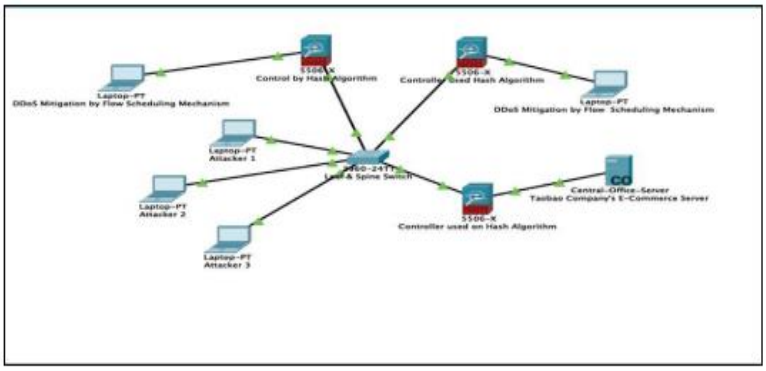


Figure (05) : Flow Scheduling Topology

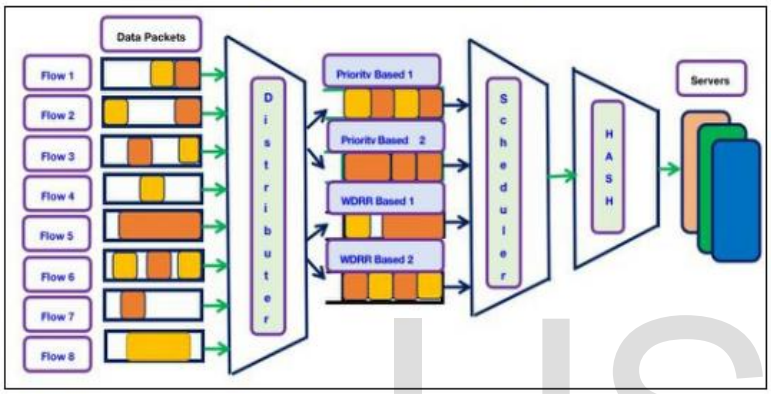


Figure (06) : Flow Scheduling Mechanism

End-host implementation: FlowSM priority-based packet scheduling needs to extend all the way to the end-host to be fully effective. In fact, we think of the FlowSM as starting at the NIC and in our simulations we assume that the NIC queues also implement Flow Scheduling priority scheduling or dropping mechanisms. An alternative design may push the contention to software queues by rate-limiting the traffic to the NIC. Priority scheduling can then be implemented in software across active flows. This approach does not require NIC changes and also avoids dropping packets at the end-host but it requires more sophisticated software particularly at 10Gbps speeds. The reader may also wonder about the feasibility of our rate control implementation. Specifically, our rate control frequently operates at line rate and uses a fixed re-transmission timeout value typically set to 3×RTT which can be quite small. However our simulations show that the timeout can be set to larger values in practice without impacting performance.

Hash Function Implementation: A hash function may be considered to perform three functions, these are Convert variable length keys into fixed length values by folding them by words or other units using a parity-preserving operator like ADD or XOR. Scramble the bits of the key so that the resulting values are uniformly distributed over the key space.It should be very fast to compute. It should minimize duplication of output collisions value. Map the key values into ones less than or equal to the size of the table.

Detecting and Mitigating DDoS Attacks: Hash functions rely on generating favorable probability distributions for their effectiveness, reducing access time to nearly constant. Hash functions can be designed to give best worst-case performance good performance under high table loading factors and in special cases and perfect mapping of keys into hash codes. Implementation is based on parity-preserving bit operations (XOR and ADD), multiply or divide. A necessary adjunct to the hash function is a collision-resolution method that employs an auxiliary data structure like linked lists or systematic probing of the table to find an empty slot. When testing a hash function the uniformity of the distribution of hash values can be evaluated by the chi-squared test. This test is a goodness-of-fit measure that it's the actual distribution of items in buckets versus the expected or uniform distribution of items. The formula are

$$\frac{\sum_{n=0}^{m-1}(bj)(bj+1)/2}{n/2m(n+2m-1)}$$

1. $p_{Ires} = p_I(t) + \min_{I \in S(I)} (Ui'(xi(t)) - \sum_{k \in L(I)} p_k(t)) / |L(i)|$
2. $p_I^{new} = [p_I^{res} - \eta (1 - (\sum_{I \in S(I)} xi(t))) / c_I \cdot p_I(t)]$
3. $P_I(t+1) = \beta p_I(t) + (1-\beta) p_I^{new}$

Finally, where size-based traffic prioritization may reduce the stability region of the network. The stability is in the stochastic sense meaning that the network may be unable to keep up with flow arrivals even though the average load on each link is less than its capacity. However, this problem is mostly for “linear” typologies with flows traversing different numbers of hops intuitively. It is due to the trade off between priority based small flows versus maximizing service parallelism on long routes. We have not seen this issue in our research study and do not expect it to be a major concern in real data center environments because the number of hobs is very uniform in data center fabrics and overall load contributed by the small flows is small for realistic traffic distributions.

6. Evaluations:

In this section, we present to deconstruct of overall the results and to demonstrate weight deficit flow scheduling mechanism to detect and mitigate the large number of DDoS attacks that contribute to the performance. Over subscription of network links out of spine-leaf to core should also be considered an extensive ns2 simulation based evaluation of weight deficit flow scheduling mechanism.

Simulation Methodology: We show how the achieves network are more optimal end-to-end performance in realistic data center networks running workloads that have been observed in deployed data centers. Finally, we have to deconstruct the overall results and demonstrate the factors that contribute to the performance. we evaluate flow scheduling mechanism of performance using extensive packet-level simulations in the ns2 network simulator.

- Fast convergence to optimal allocation in dynamic flow based on Priority and Deficit Weight.
- Flexibility and Reliability for meeting various bandwidth allocation objectives.
- Bandwidth become faster and more reliable by using WDRR flow scheduling mechanism.
- Detecting and mitigating to DDoS attackss in real time by using of hash block.

Flow Scheduling Topology: We use the leaf-spine topology shown in Figure 10,11. The flows are interconnected in 16 hosts through 4 leaf switches and also connected to 4 spine switches in a full mesh. Each leaf switch has 12 10Gbps down-links (to the hosts) and 4 40Gbps up-links (to the spine) resulting in a non-oversubscribed flow scheduling.

Scheduling load-balancing: We use packet spraying where each switch sprays packets among all shortest-path next hops in round- robin fashion. We have also experimented with Equal Cost Multipathing (ECMP) which hashes entire flows to different paths to avoid packet reordering. Overall, we found that for all schemes, the best results are obtained with packet-spraying after fast re-transmissions are disabled to cope with packet reordering . In fact, this is the reason we disabled 3 dupACKs in our rate control. Hence, we use packet spraying by default for all schemes.

Congestion Feedback: Flow scheduling mechanism uses a feedback loop between the source and destination in leaf switches and spine switches to popular the remote metrics in the Congestion-To-Leaf Table at each leaf switches. We could describe the sequence of the event involved part. The source leaf sends packets to the switches with including tag field set to the up-link port taken by the packet. It also sets on CE field up on the price or cost or link distributed dynamic weight sharing schemes to detecting the Trojan Virus or detecting and malicious attack on those links to reduce the congestion, detecting and mitigating a large number of DDoS attacks.

Flow-let Detection: Flow-lets are detected tracked in the leaf switches using the Flow-let figure 10. Each entry of the table consists of the port number a valid bit and an age bit. These are such as

- If the entry is valid, the Flow-let is active and the packet is sent on the port indicated in the entry.
- If the entry is not valid, the incoming packet starts a new Flow-let. In the case, we make a load balancing decision and cache the result table for using by the subsequent packets.

Each incoming packet reset the age bit. If the age bit is set when the timer checks it and then it ensured that there have no any packets for the entry in the last seconds and entry times out.

Load Balancing Decision Logic : Load balancing decisions are made on the first packet of each Flow-let. For a new Flow-let, we pick up-link port that minimize to maximum of the local metric and the remote up-links are good equality. The one is chosen at random with preference given to port cached in the entry in Flow-let. It's flow only moves if there is some strictly priority better up-link than the one its last Flow-let links. We repeat the previous simulation but stress the network by using up to 50 concurrent large flows to a single destination port and measure the overall loss rate. We conduct the simulation both with and without flows probe mode. This is because except for the high-priority flow, the packets of the other flows are all dropped at the bottleneck. Hence, each low-priority flow re-transmits a full sized (1500B) packet every RTO = 45μs which is eventually dropped. As expected, the probe mode significantly lowers the loss rate since the low priority flows only periodically send a small probe packet while waiting for the high priority flow to complete. We can show flow scheduling performance for In-cast traffic patterns which occur in many large-scale web applications and storage systems and have shown to result in throughput degradation for TCP. The senders respond with 100MB/N of data simultaneously. The request completes when all the individual flows have finished. Once a request is complete, the client immediately initiates the next request. The simulation is run for 1000 requests that we compute the average total request completion time and the average individual flow completion times. Hence, considering the total request completion time, all schemes handle In-cast fairly well. DCTCP does the best and achieves a near-ideal request complete time of 80ms across all number of senders. Flow Scheduling Mechanism is almost as good achieving a total request completion time of 81.1ms at 50 senders. The small increase is due to the slight overhead of serially scheduling flows. However, as expected serial flow scheduling significantly improves the av-

erage individual flow completion times for scheduling compared to DCTCP and TCP - Drop Tail which are more optimal fair into across flows. PDQ also exhibits a similar behavior as weight deficit flows since it aims to the same kind of flow scheduling. However it has slightly higher overhead in flow switching and consequently shows slightly worse performance as the number of flows increases.

We have explored a number of other scenarios including weight deficit flow scheduling performance in over subscribed topology and with different load-balancing mechanisms. The importance of the starvation prevention mechanism and the implications of having weight flows into different switches only at leafs and not in the core. In all these mechanism, we have found that scheduling mechanism achieves very good performance.

Functionality of Available Bandwidth Monitoring:

Mitigating the DDoS Attacks: I want to construct more sophisticated and effective available bandwidth to mitigate the large scale of DDoS attacks. By this purpose we should claim that the client information would able to more sophisticated and more secure from their destination server to client server communication.

Signature based Detection Mechanism Mitigate to DDoS attacks: Signature based mechanism can easily detecting the intruder and provide more secure to data communications . There are so many types of signature based mechanism. All are the most common thing is that public key and their private key which make your data communication provide more secure and preventing the DDoS attackss which can be detected with signature based detection. In signature based detection mechanism, we are basically trying to create a profile or a signature associated with a particular kind of attack which is saved in our database for matching when we encounter an attack. However, the number of types of DDoS attacks are increasing day by day starting with the typical TCP flooding, UDP flooding, ICMP flooding, SYN flooding and certain source based and destination based bandwidth and scanning attack with the evolution of new next generation networks. The new next generation attacks are also evolving which are quite hybrid in nature and difficult to catch when we try to match them against some set signature profiles. Besides,it becomes an even bigger challenge when we consider distinguishing DDoS attacks from legitimate large volumes of traffic which we term as Flash Crowds.

FlowSM Coordinate Mitigate to DDoS Attacks: Now the main solution remains in creating the right profiles based on deep anomaly analysis which has a necessity to be real time and also which is capable of evolving just as our evolving attack types. Distributed Denial of Service (DDoS) is a type of attack in which the attacker tries to degrade the performance of server or network so that the 2/3 servers may not provide service to legitimate users. Since there is a huge increase in DDoS attacks which has created many financial losses in the E-Commerce world. To avoid the losses incurred because of DDoS attacks, efficient mechanisms are required to counter these attacks. In the proposed approach routers collectively try to mitigate the DDoS attacks on the server. There are three steps in the different proposed approaches. Initially, for attack detection and classification destination router which is attached to the victim or monitors continuously the traffic pattern. Secondly, once the attack is detected destination router tries to balance the load using the NAT (Network Address Translator). Thirdly, whenever the attack is detected to mitigate different types of attacks. The signature is push back to upstream routers so that the upstream routers start monitoring the traffic and apply the mitigation mechanism depending on type of attack detected. In this paper, we use digital signature mechanism to encrypt our data transmission which has two type of key generation that are public key and private key. Both sender and receiver have their own public key . It's a common key to generate for using of sending their data and using their receiver private key to encrypt their data. Finally that data could not detect by attacker and then receiver open that using their private key. This algorithm makes to us secure communication in a organization.

7. ANALYSIS:

Global Congestion Awareness: Handling asymmetry essentially requires non-local knowledge about downstream congestion at the switches with asymmetry. A switch cannot simply balance traffic based on the congestion of its local links. In fact, this may lead to even worse performance than a static scheme such as ECMP because of poor interaction with TCP's control loop. As an illustration, we consider the simple asymmetric scenario in Figure 10, 11. Leaf L0 has 100Gbps of TCP traffic demand to Leaf L1. Static ECMP splits the flows equally achieving a throughput of 90Gbps because the flows on the lower path are bottle necked at the 40Gbps link (S1, L1). Local congestion-aware load balancing is actually worse with a throughput of 80Gbps. This is because as TCP slows down the flows on the lower path the link (L0, S1) appears less congested. Hence, paradoxically, the local scheme shifts more traffic to the lower link until the throughput on the upper link is also 40 Gbps. In-fact, global congestion-aware load balancing does not have this issue. The reader may wonder if asymmetry can be handled by some form of oblivious routing such as weighted random load balancing with weights chosen according to the topology. Note that, in

this example such as the two LS1(Leaf Swicth 1)→LS2(Leaf Swicth 2) paths are symmetric when considered in isolation. But because of an asymmetry in another part of the network the LS0(Leaf Swicth 0)→LS2(Leaf Swicth 2) traffic creates a bandwidth asymmetry for the LS1(Leaf Swicth 1)→LS2 (Leaf Swicth 2) traffic that can only be detected by considering non-local congestion.

Congestion Feedback: Each packet carries a congestion metric in the overlay header that represents the extent of congestion the packet experiences as it traverses through the fabric. The metric is updated hop-by-hop and indicates the utilization of the most congested link along the packet's path. This information is stored at the destination leaf on a per source leaf, per path basis and is opportunistically feed back to the source leaf by piggybacking on packets in the reverse direction. There may be in general 100s of paths in a multi-tier topology. Hence, It reduces to state that the destination leaf aggregates congestion metrics for one or more paths based on a generic identifier called the Load Balancing Tag that is source leaf inserts in packets.

Resource Pooling: The goal of resource pooling is to make a collection of network links behave as though they make up a single link with the aggregate capacity. This is useful in data centers where data has a large number of paths and we would like flows to use the entire pool of capacity efficiently. The Multipath TCP (MPTCP) congestion control algorithm has recently been proposed for achieving resource pooling. MPTCP divides a flow into sub-flows that traverse different paths and implements a coordinated congestion control across them to realize resource pooling. The turns out that resource pooling for multipath flows can expressed as the flow scheduling mechanism problem to detect the DDoS attacks. The key idea is to consider the utility for a flow in terms of the total rate of all its sub-flows. Any sharing or any fairness objective can be generalized for multipath resource pooling by this way.

Priority dropping: Whenever a packet arrives to a port with a full buffer. If it has priority less than or equal to the lowest priority packet in the buffer then it is dropped. Otherwise, the packet with the lowest priority is dropped to make room for the new packet. That time we can use Weight Deficit Round Robin flow scheduling mechanism to give them a Quantum ratio for every round circle with their Quantum base. If match then check Quantum Ratio and Quantum base run to around of whole process and else adding with quantum base for checking others packet of network. In this way, we can get bandwidth faster, more reliable and minimizing the congestion time. Strict Priority is the distributed network that obtained fast bandwidth, reduce feedback delay, reduce cost of base station synchronization, efficient feedback compression, low phase noise transmitters and congestion-aware load balancing mechanism for mitigating the large number of DDoS attacks.

Find Maximum Deficit Weight of all Links & Find Minimum Deficit Weight of Links : In WDRR queuing packets are first classified into various service classes such as real time, interactive and file transfer and then assigned to a queue that is specifically dedicated to that service class. Each of the queues is serviced in a round robin order. Similar to strict priority queueing and fair queuing empty queues are skipped. WDRR queuing is also referred to custom queuing. WDRR queuing supports the allocation of different amounts of bandwidth to different service class by either:

- Allowing higher-bandwidth queues to send more than a single packet each time that are visited during a service round.
- Allowing each queue to send only a single packet each time that it is visited but to visit higher-bandwidth queues multiple times in a single service round.
- Weight Deficit round robin is a new variation of WRR that achieves better GPS approximation with knowing the mean packet size and weight of each connection in advance.

To Mitigate DDoS Attacks: When hashing strings, it is not uncommon for more than one unique string to end up with the same hashed value. This is known as a Hash Collision. Normally this is not a problem, as languages use various strategies to resolve these collisions. However, an attacker can exploit this behavior to exhaust the CPU by forcing a large number of collisions via a single request with many parameters. The core issue has seen on this research paper and as a result several languages such as Perl and Ruby, have added randomization to their hash functions. Other languages have recently limited the number of parameters allowed in a single request such as PHP's max_input_vars directive. By default, the max_input_vars directive limits the number of GET, POST and Cookie input variables to 1000. For those who can not patch their application framework, Mod Security can be used to virtually patch this flow. However the swift language is more develop than these languages. So it is the first time to use the hash algorithm to mitigate the DDoS attacks by the Swift language. Using Swift try to find attacker and mitigate to large-scale of DDoS attacks.

Flow-let size: The distribution of the data bytes versus flow-let size for three choices of flow-let inactivity gap: 250ms, 500μs, and 100μs. Since it is unlikely that we would to see a gap larger than 250ms in the same application-level flow, the line "Flow (250ms)" essentially corresponds to how the bytes are spread across flows. The plot shows that balancing flow-lets gives significantly more fine-grained control than balancing flows. Even with an in activity gap of 500μs which is quite large and

poses little risk of packet reordering in data centers, we see nearly two orders of magnitude reduction in the size of transfers that cover most of the data: 50% of the bytes are in flows larger than ~30MB, but this number reduces to ~500KB for "Flow-let (500μs)".

In this research paper, We can proposes a novel cooperative technique that is based on a feedback-control strategy. Our scheme tackles DDoS attacks in different consecutive phases namely like as control phase, negotiation phase, stabilization phase and processing phase. This technique overcomes all challenges that we enumerated above. The performance of the scheme over previous techniques are as follows. It does not require universal deployment. It can be implemented in the today inter infrastructure.

8. METHODS:

In this research paper, we are using malicious node on easily identified and routing protocol become security aware. Each node provides identification malicious nodes this because DSR does not have any blacklist for sensor networks. Neighbor node send request to other neighbor node and then nearby neighbor node send the request continuously carried on until the traffic arrive the destination node. By this purpose, flow scheduling mechanism is being compare to their performance of before using and after using flow scheduling mechanism prevent some methods.

Make Priority of router with the minimum hop count:

- If multiple OSPFv2 processes learn routes to the same destination and the external and internal priorities of the routes are the same then the system selects to the route within the smallest link cost.
- If the link costs of the routers are the same then the router participate into load balancing.
- If multiple OSPFv3 processes learn routes to the same destination and the external and internal priorities of the routes are the same then the system selects within the smallest process ID.
- If multiple IS_IS processes learn routes to the same destination and the external and internal priorities of the routes are the same then the system selects the route within the smallest link cost.
- If the link costs of the routes are the same then the routes participate in load balancing.
- If multiple RIP or RIPng processes learn routes to the same destination and the external and internal priorities of the routes are the same then the system selects the route within the smallest link cost.
- If the link costs of the routes are the same then the routes participate in load balancing.

Coordinate malicious nodes flooding packets to the destination nodes: Network technology is moving towards changing the wired connection between nodes in the network to a wireless connection which makes the network more flexible. There are two types of wireless networks such as infrastructure networks and infrastructure less networks. In infrastructure networks, nodes depend on a central node to coordinate the communication between them. But in infrastructure-less networks, nodes depend on themselves to coordinate the communication process. Mobile Ad Hoc Network (MANET) is an infrastructure less network that connects mobile nodes via wireless links like radio and microwave signals.

DCN Build Using a Leaf Spine Architecture: A traditional three-tiered model was designed for use in general networks usually segmented into pods which constrained the location of devices such as virtual servers. The architecture consists of Core Routers, Aggregation Routers, and Access Switches. These devices are interconnected by pathways for redundancy which can create loops in the network. As part of the design, a protocol (Spanning Tree) that prevents looped paths is implemented. However, it is doing so deactivates all but the primary routes. A backup path is only brought up and utilized when the active path experiences an outage.

Dynamic Weight Flow Scheduling Algorithm: Dynamic flow schedule mechanism make to maximize network throughput and to minimize the conflict between elephant and mice flows. CDFS schedules elephant flows with a scalable "Max-min Weight" scheme, which is adaptive to the available bandwidth of links. Thus switches route mice flows with multipath routing method. Experiment results show that CDFS improves network throughput by 10% when mice flows dominate the network traffic, and reduces mice flow latency by at least 60% compared to the Flow scheduling mechanism model.

9. EXPERIMENTS:

In this research paper, we propose that dynamic weight flow scheduling mechanism that every node has a weight and capacity. The node can not exceed to their capacity. Therefore, we find the maximum possible flow to overcome their unfair utilization link of bandwidth, provides maximum fair utilization link of bandwidth and produce the better link utilization with moderate additional cost. Then minimizing the FCT congestion control to make flexibility to implement on any network resource allocation. After that we modify the utility of function and also the bandwidth of function to consider the aggregate throughput achieved by the weight flow scheduling algorithm. The running time of max-min weight flow scheduling algorithm is linear with the increasing of network nodes and maximum link weight flows. Dynamic flow scheduling performs better than open flow and ECMP mechanism in network throughput when mice flows dominate the network traffic. That Max-Min weight algorithm avoid to flow completion time and make more available bandwidth that would be faster and more flexible. By this flow scheduling algorithm, we can achieve faster and more flexible bandwidth that would be able to mitigate the large scale of DDoS attacks.

Firstly, we need to make a good design with based on NS2 network simulator and then constructed novel distributed algorithm is Dynamic Weight Flow Scheduling Algorithm to make that bandwidth equally distributed to other mechanism.

Secondly, we need to implement hash algorithm to make the distributed bandwidth faster and also make it secure by using Swift language and network frame encode by that mechanism.

Thirdly constructed Flow let Completion Time (FCT) to make their bandwidth much more efficiency with the help of RTT and EWMA filter and using DGD algorithm calculate link optimal weight flow cost and estimate to fix their total cost of efficiency.

Simulation Time in (seconds)	Number of Connections Generated Flow Scheduling	Traffic Name	Strict Priority Flow Scheduling Mechanism		
			High	Medium	Low
01	32	DCTCP	12	10	10
30	32	TCP Drop Tail	12	10	10
60	32	PDQ	12	10	10
90	32	PFabric	12	10	10
120	32	NUMFabric	12	10	10
150	32	FlowSM1	12	10	10
180	32	FlowSM2	10	10	10

Figure (07): Number of connections admitted schemes with priority flow scheduling (Table)

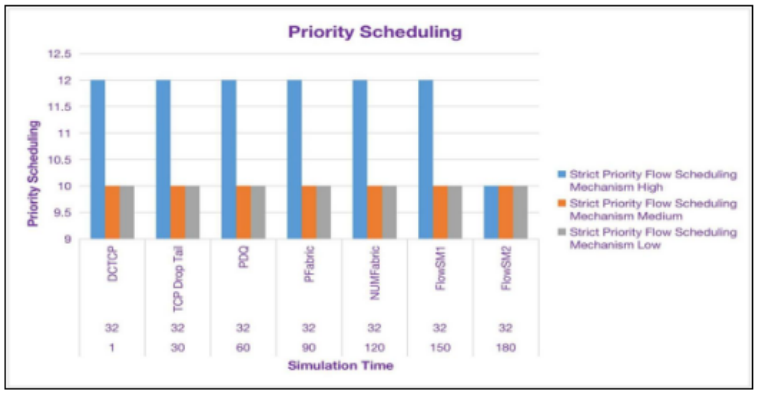


Figure (08): Number of connections admitted schemes with priority flow scheduling

Finally, we filter the network by hash function on bandwidth which provide to the low bypass solution as the mean of low latency efficiency. As therefore, we can get more fast efficiency bandwidth and securely to communication to other host that had never been constructed in before. As end host feel that their information and documentation will more secure and using of bandwidth will faster and more flexible and secure. For these purpose, the attacker will never get any information from servers.

Traffic Stage	Delay Time	Jitter Time	Respose Time	Loss	Bandwidth Allocation
VoIP	<150 ms	<30 ms	----------	<0.5%	High
FTP	Med	None	2-3 sec	<0.13%	High
HTTP	<400 ms	None	2-3 sec	<0.07%	40<10Gbps
Email	Low	None	2-4 sec	<0.023%	40<10Gbps

Figure (09): QoS requirements for the supported types of traffic

Scheme Number	Weight Q1	Weight Q2	Weight Q3	Weight Q4	Maximum gap of BW
Scheme 1	0.13	0.17	0.23	0.33	12%
Scheme 2	0.15	0.27	0.33	0.45	17%
Scheme 3	0.16	0.25	0.35	0.47	17%
Scheme 4	0.17	0.35	0.45	0.55	19%
Scheme 5	0.18	0.37	0.47	0.57	15%

Figure (10): Distributes Weight Flow Scheduling for the supported types of traffic (Table)

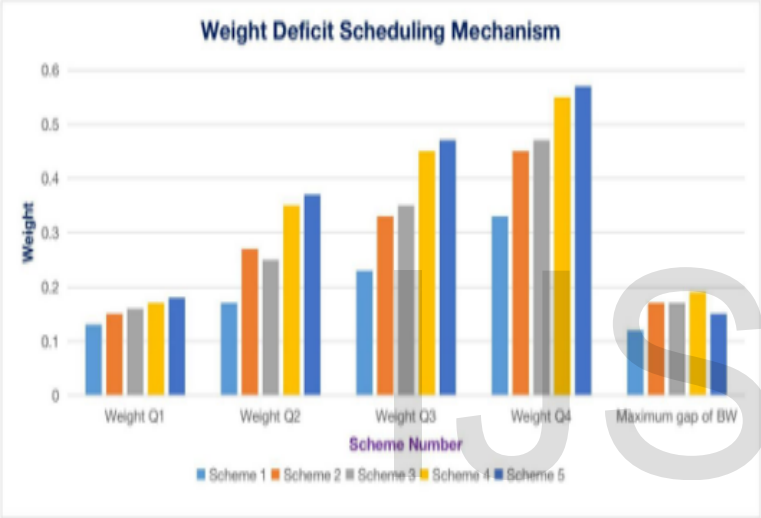


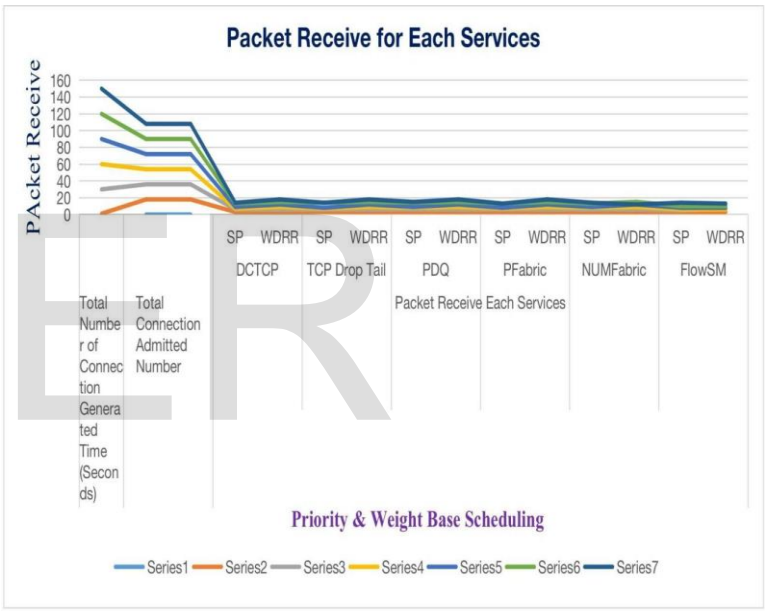
Figure (11): Distributes Weight Flow Scheduling for the supported types of traffic

Name of Scheme Scheduling	Packet Size	init Cwnd Size	RTO
DCTCP	225KB	12 pkts	220
TCO Drop Tail	225KB	12 pkts	220
PDQ	225KB	12 pkts	220
PFabric	36KB	12 pkts	45
NUMFabric	225KB	12pkts	220
FlowSM (WDRR)	180KB	12pkts	45

Figure (12): Scheme Scheduling Packet RTO Table

Total Number of Connection Generated Time (Seconds)	Total Connection Admitted Number		Packet Receive Each Services											
			DCTCP		TCP Drop Tail		PDQ		PFabric		NUMFabric		FlowSM	
	SP	WDRR	SP	WDRR	SP	WDRR	SP	WDRR	SP	WDRR	SP	WDRR	SP	WDRR
01	18	18	3	3	3	3	3	3	3	3	3	3	3	3
30	36	36	6	6	6	6	6	6	6	6	6	6	5	7
60	54	54	7	9	7	9	8	9	7	9	8	9	6	6
90	72	72	9	12	8	12	9	12	8	12	9	12	8	8
120	90	90	13	15	14	15	14	15	13	15	13	15	10	10
150	108	108	14	18	14	18	15	18	13	18	14	12	14	13

Figure(13): Number of connections admitted for varying number connections



Figure(14): Number of connections admitted for varying number connection

10. Discussions:

Weight deficit flow scheduling mechanism is more generally advocates a different design philosophy for data center networks. In our thought process is informed by the fact that the data center network is more an interconnect for distributed computing workloads rather than a bit-pipe. Hence we believe that it is more important to the network resource allocation to meet overall computing objectives rather than the traditional communication metrics such as throughput and fairness which TCP optimizes. We discuss some other common concerns that might come up with a design like Weight Deficit Round Robin Algorithm.

Flow Scheduling Starvation: A potential concern with strictly priority based small flows may starve large flows. Moreover, a malicious user may game the system by splitting up to large flows to gain an advantage. In-fact, these issues are being an unique to flows any system that implements WDRR like scheduling has these concerns. WDRR actually improves the majority of flows compared to TCP's fair sharing. The intuition is that for heavy-tailed distributions, small flows contribute a small fraction of the overall traffic. Hence the strict priority based on them has little impact on the large flows and in fact helps them because they complete quickly which reduces network contention.

Scheduling Setting Packet Priorities: In many data center applications flow sizes or deadlines are known at initiation time and can be conveyed to the network stack to set priorities. In other cases, we expect that scheduling mechanism would achieve good performance even with imprecise but reasonable estimates of flow sizes. Within the realistic distributions most of the benefit can be achieved by classifying flows into a few priority levels based on size. The intuition is that it suffices that at any instant at each switch the priority dequeueing order is maintained.

Supporting multiple priority schemes: In practice, data center fabrics are typically shared by a variety of applications with different requirements and a single priority scheme may not always be appropriate. This can easily be handled by operating the strict priority scheduling and dropping mechanisms within individual "higher-level" traffic classes in an hierarchical fashion. Traditional QoS mechanisms such as WRR are used to divide bandwidth between these high-level classes based on user defined policy, while the mechanism provides near-optimal scheduling of individual flows in each class according to the class's priority scheme.

Stability: Finally, the theoretical literature has demonstrated scenarios where size-based traffic prioritization may reduce the stability region of the network. Here, stability is in the stochastic sense meaning that the network may be unable to keep up with flow arrivals even though the average load on each link is less than its capacity. However, this problem is mostly for "linear" topology with flows traversing different numbers of hops intuitively it is due to the trade off between prioritizing small flows versus maximizing service parallelism on long routes. We have not seen this issue in our research level and do not expect it to be a major concern in real data center environments because the number of hops is very uniform in data center networks and the overall load contributed by the small (high-priority) flows is small for realistic traffic distributions.

11. CONCLUSIONS:

In this paper, We proposed an algorithm solution for mitigating the large scale of DDoS attacks. As the various parameters varies in the DDoS attacks that should able to detect and handle to get an amount of vulnerable traffic. We have an emphasis on dynamic weight flow scheduling algorithm that are more sophisticated to get more pinged link for DDoS attacks. To handle the DDoS attacks, we check as the weight flow scheduling mechanism with fast and flexible bandwidth to mitigate the large scale of DDoS attacks. At the holding point of view, we can provide the more fast and more flexible and more secure bandwidth provides to client to mitigate the large scale of DDoS attacks.

Many applications in data centers drop into three broad categories which are Bandwidth-consuming, Latency-Sensitive and Detecting Flow Congestion Control. Bandwidth-consuming applications generate long lived and high-throughput flows used to mitigate the DDoS attacks. While latency sensitive applications produced mice flows with relatively small size. Elephant flows compromise most of the network throughput, whereas a majority of flows are in mice ones and detecting flow congestion. One of the main challenges for cooperative techniques is how to find location of attackers and then rate limit traffic at point close to the source of attacks. To achieve this goal, flow scheduling mechanism techniques based on Max-Min Weight Flow Algorithm have been proposed to find location of attackers. These technique had assumed to mitigate the DDoS attacks and should improve the overall maximum bandwidth throughput and avoided to flow congestion. Therefore, the path with available bandwidth is more optimal. This observation motivates us to develop an effective mechanism for both types of flow. We had assumed that the state of the art elephant flow scheduling mechanism was beneficial to improve the network throughput. Thus, It should make every effort to avoid conflicts between elephant and mice flows. Moreover, with the low dependency on the monitoring system our proposed mechanism can simplify the deployment and save the energy consumption. Recent technologies merely handle elephant flows or only resolve latency issues. In other contrast, our work is devoted to handling both type of flows.

12. FUTURE WORKS:

In this research paper, we presents that dynamic weighted flow scheduling mechanism to mitigate the large scale of DDoS attacks. This flow overcome the unfair utilization link of bandwidth and provides maximum fair utilization link of bandwidth and produce the better link utilization with moderate additional cost that the presented scheme successfully does its work avoiding saturation of core , leaf, spine switches and to mitigate the large scale of DDoS attacks. Thus this research paper is the best paper that had never done in

before. Considering more number of parameters would certainly improve the efficiency and robustness of proposed approach. Therefore identification and incorporation of additional parameters for the detection of DDoS attacks are planned as a future work. Moreover, the evaluation of proposed approach on flow scheduling mechanism.

13. Acknowledgment:

I would like to more thank to my most honorable supervisor who give me a great chance to work this project and give me much more advice and information to improve this research paper. Moreover, I deeply appreciate to him to give me his valuable time and always devoted to me, guided me, to despite his extremely busy schedule time and his rigor has been a great value in this work. I would also like to thank to reviewer whose comment is more efficient and helped us to improve this research paper. This paper would be the greatest paper that had never done it before.

14. References:

1. R. Mahajan,S.M.Bellovin, S. Floyd, Controlling high bandwidth aggregates in the network, ACM SIGCOMM Computer Communication Review 32 (3) (2002) 62–73.
2. D. Y. Yau, J. C. S. Lui, F. Liang, Y. Yam, Defending against Distributed Denial-of-Service Attacks with Max-Min Fair Server- Centric Router Throttles, IEEE/ACM Transactions on Networking 13 (1) (2005) 29–42.
3. Li Chen, Kai Chen, Wei Bai, Mohammad Alizadeh(MIT), Scheduling Mix-flows in Commodity Data-centers with Karuna,Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference, ACM, 2016.
4. Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, Masato Yasuda, Less is more: trading a little bandwidth for ultra-low latency in the data center, Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), 2012.
5. Andrew R. Curtis Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, DevoFlow: scaling flow management for high-performance networks, ACM SIGCOMM Computer Communication Review 41.4 (2011): 254-265.
6. Al-Fares, Mohammad, Alexander Loukissas, Amin Vahdat, A scalable, commodity data center network architecture, ACM SIGCOMM Computer Communication Review, Vol. 38. No. 4. ACM, 2008.
7. de Oliveira, R. L. S., Schweitzer, C. M., Shinoda, A. A., Prete, L. R., Using mini net for emulation and prototyping software-defined networks, Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on. IEEE, 2014.
8. Information on:http://stackoverflow.com/questions/18552964/finding-path-with-maximum-minimum-capacity-in-g Caph.
9. M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, Hedera: Dynamic Flow Scheduling for Data Center Networks. In Proc. NSDI, Apr. 2010.
10. Hong, Chi-Yao, Matthew Caesar, P. Godfrey, Finishing flows quickly with preemptive scheduling, ACM SIGCOMM Computer Communication Review 42.4 (2012): 127-138.
11. Sonia Laskar.Qualified Vector Match and Merge Algorithm (QVMMA) for DDoS Prevention and Mitigation [J] Procedia Computer Science 79 (2016) 41 - 52
12. Bing Wang.DDoS attacks protection in the era of cloud computing and Software-Defined Networking [J] Computer Networks 81 (2015) 308-319
13. Jian Zhang, Yawei Zhang, Pin Liu, and Jianbiao He. A Spark-Based DDoS attacks Detection Model in Cloud Services, pages 48-64. Springer International Publishing, Cham, 2016.
14. Zuo Qingyun.Study on SDN technology based on Open Flow [J], Chinese Journal of Software, 2013,24 (5): 1078-1097
15. M.Al-Fares,S.Radhakrishnan,B.Raghavan,N.Huang,and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In NSDI, 2010.
16. M. Alizadeh et al. pFabric: Minimal Near-optimal Data-center Transport. In SIGCOMM, 2013.
17. M.BeckandM.Kagan.PerformanceEvaluationoftheRDMAover Ethernet (RoCE) Standard in Enterprise Data Centers Infrastructure. In DC-CaVES, 2011.
18. S. Kandula, D. Katabi, S. Sinha, and A. Berger. Dynamic Load Balancing Without Packet Reordering. SIGCOMM Comput. Commun. Rev., 37(2):51–62, Mar. 2007.
19. S.Kandula,S.Sengupta,A.Greenberg,P.Patel,andR.Chaiken.The nature of data center traffic: measurements & analysis. In IMC, 2009.
20. M.Kodialam,T.V.Lakshman,J.B.Orlin,andS.Sengupta. Oblivious Routing of Highly Variable Traffic in Service Overlays and IP Backbones. IEEE/ACM Trans. Netw., 17(2):459–472, Apr. 2009.
21. Alizadeh et al. pFabric: Minimal Near-optimal Datacenter-Transport. In SIGCOMM, 2013.
22. M.Alizadehetal.CONGA:Distributedcongestion-awareloadbalancing for Data-centers. In SIGCOMM, 2014.
23. D.Bickson,Y.Tock,A.Zymnis,S.P.Boyd,andD.Dolev.Distributed large scale network utility maximization. In ISIT, 2009.
24. P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown,M. Izzard, F. Mujica, and M. Horowitz. Forwarding Metamorphosis: Fast Programmable Match-action Processing in Hardware for SDN. In SIGCOMM, 2013.
25. M. Chowdhury and I. Stoica. Efficient coflow scheduling without prior knowledge. In SIGCOMM, 2015.
26. M. Chowdhury, Y. Zhong, and I. Stoica. Efficient Coflow Scheduling with Varys. In SIGCOMM, 2014.

27. C.-Y. Hong, M. Caesar, and P. B. Godfrey. Finishing Flows Quickly with Preemptive Scheduling. In SIGCOMM, 2012.

28. L. Jose et al. High Speed Networks Need Proactive Congestion Control. In HotNets, 2015.

29. D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In SIGCOMM, 2002.

30. S. Keshav. A Control-theoretic Approach to Flow Control. In SIGCOMM, 1991.

31. K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti. NUMFabric: Fast and Flexible Bandwidth Allocation in Data-centers. https: //people.csail.mit.edu/alizadeh/papers/numfabric- techreport.pdf.

32. D. P. Palomar and M. Chiang. Alternative distributed algorithms for network utility maximization: Framework and applications. IEEE Transactions on Automatic Control, 52(12):2254–2269, 2007.

33. L. Popa et al. Faircloud: sharing the network in cloud computing. In SIGCOMM, 2012.

34. D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In NSDI, 2011.

35. M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In Proc. of SIGCOMM, 2008.

36. M.Al-Fares,S.Radhakrishnan,B.Raghavan,N.Huang,andA.Vahdat. Hedera: dynamic flow scheduling for data center networks. In Proc. of NSDI, 2010.

37. M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center TCP (DCTCP). In Proc. of SIGCOMM, 2010.

38. M.Alizadeh,A.Kabbani,T.Edsall,B.Prabhakar,A.Vahdat,andM.Yasuda. Less is more: trading a little bandwidth for ultra-low latency in the data center. In Proc. of NSDI, 2012.

39. C.Raiciu,S.Barre,C.Pluntke,A.Greenhalgh,D.Wischik,andM.Handley. Improving datacenter performance and robustness with multipath TCP. In Proc.of the SIGCOMM, 2011.

40. B.Vamanan,J.Hasan,andT.N.Vijaykumar.Deadline-AwareDatacenterTCP(D2TCP). In Proc. of SIGCOMM, 2012.

41. V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective fine-grained TCP retransmissions for datacenter communication. In Proc. of SIGCOMM, 2009.

42. M.Verloop,S.Borst,andR.Nu n ezQueija.Stabilityofsize-basedscheduling disciplines in resource-sharing networks. Perform. Eval., 62(1-4), 2005.

43. C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. Better never than late: meeting deadlines in datacenter networks. In Proc. of SIGCOMM, 2011.