CAS 781: Data Center Design - Final Project

Baran Kaya, 400284996

Content

I. Papers Reviewed 1
II. Introduction 1
III. Paper Summaries 2
A. Hedera: Dynamic Flow Scheduling for Data Center Networks 2
B. Datacenter Traffic Control: Understanding Techniques and Trade-offs 3
C. Efficient Load Balancing Over Asymmetric Datacenter Topologies 5
D. Sorted-GFF: An Efficient Large Flows Placing Mechanism in Software Defined Network Datacenter 7
E. JITeR: Just-In-Time Application-Layer Routing 8
F. VARMAN: Multi-Plane Security Framework for Software Defined Networks 9
IV. Comprehensive Summary 11
V. References 12

I. Papers Reviewed

1. Hedera: Dynamic Flow Scheduling for Data Center Networks
2. Datacenter Traffic Control: Understanding Techniques and Trade-offs
3. Efficient load balancing over asymmetric datacenter topologies
4. Sorted-GFF: An efficient large flows placing mechanism in software defined network datacenter
5. JITeR: Just-in-time application-layer routing
6. VARMAN: Multi-plane security framework for software defined networks

II. Introduction

For the final project of the Data Center Design course, I've reviewed the papers listed above. My topic for this literature review was Data Center Networking. This project has both individual paper reviews and all of the papers' comprehensive review. Section 3 has each papers' respective summaries and section 4 has all the papers connection and their addition into datacenter networks. Listed papers have topics such as network load balancing/flow scheduling, data center network information and network security. Some of the papers has/use Software-Defined Network (SDN) over datacenter networks.

III. Paper Summaries

A. Hedera: Dynamic Flow Scheduling for Data Center Networks

Hedera [1] is a dynamic network flow scheduling algorithm that designed for multi-rooted tree networks. Since most of the data center networks use multi-rooted trees, Hedera is designed for this kind of network architecture. Hedera's algorithm is similar to early Software-Defined Network model. It uses central scheduler which is aware of the network tree. This central scheduler monitors the whole network. When the network bandwidth exceeds the predetermined threshold value, it calculates new paths for some of the flows. The main goal of Hedera is decreasing the network tree's bisection bandwidth.

Hedera is a dynamic flow scheduling algorithm. Thus, authors compare it with static scheduling/load balancing algorithms. They mention 2 different static load balancing methods. One of them is Equal Cost Multi-Path (ECMP) and the other one is Valiant Load Balancing (VLB). Since ECMP is more common than the VLB, they used ECMP for scheduling performance comparison. They also added Non-blocking switch into comparison results for optimal/best results.

Paper starts with basic multi-rooted network architecture explanations. It describes how multi-rooted trees are used and why most of the data centers chose it over other network models. Authors also describe and show a model of their simulation and test benchmark's tree models. Then, paper explains how ECMP's and static load balancing algorithms' performance are related to the number of flows. The more the network has flows, the more static algorithms' performance is decreasing. Then, authors mention Hedera's architecture and its design. The first thing they mention is central scheduler needs 2 different algorithms for 2 different tasks. These tasks are estimating the network's actual bandwidth and finding new paths for flows.

First task of the Hedera is monitoring the network and calculating all flows actual bandwidths. Since flow's bandwidth doesn't necessarily show the flow's real demand, they created their own demand estimation algorithm. This algorithm gets source to destination N x N matrix as an input and calculates the estimated demands for given flows. They mention estimation algorithm's time complexity is O(|F|). After the estimations, central scheduler has to find new paths for some of the flows. For this, authors compare 2 different placement algorithms. These 2 algorithms are Global First Fit (GFF) and Simulated Annealing (SA). Firstly, they explain how do they use these 2 algorithms and what kind of modification they made in them. Then, they compare both algorithms with ECMP and optimal non-blocking switches.

They used Global First Fit algorithm without any modification. Because, it is already simple and efficient algorithm for finding the first suitable path for the flows. However, when the network is too crowded with flows, most of the time GFF cannot find a suitable path.

While using the Simulated Annealing, authors have to make some modifications. SA is searching the optimal solution for better paths but it requires a large search space. They overcome this problem with assigning a single core switch to each flow. This method decreases the search space of the SA. If they don't use this method, SA would take more time to calculate. Besides, longer calculation time will decrease its overall performance. Therefore, authors'

method is good for SA performance. After the modifications, rest of the section explains how Simulated Annealing algorithm works.

Later, they compare these 2 algorithms. The GFF calculates faster results than the SA but SA's calculation results are better than the GFF. So, more calculation time and complexity justifies by the better results and bandwidth gains.

After comparing the algorithms, authors explain what kind of simulator and test network they used. They tested these schedulers on 8192 and 1024 hosts simulators. They also used 16 hosts test network that they built. They used TCP flows and each test took a minute for both simulator and real world test. For better evaluation they used 3 different destination and source host selection methods. The first one is Stride(i) which sends flows to next ith host. The second one is uses tree layers' usage probability. And the last one selects destination and source hosts randomly. They used all 3 off them while evaluating the algorithms results.

Both simulator and test results show that dynamic flow schedulers are better than the static ones. All of the graphs indicate that static algorithms' performance decrease with use of upper layer switches. Also, in most cases the SA outperforms the GFF. In fact, SA results are near the non-blocking switch which is optimal. After algorithm comparison results they look at the SA iteration and calculation time data. These data show that, more iteration doesn't mean better results. They also show that, SA calculation time doesn't increase exponentially with the number of flows.

In conclusion, this paper proves that dynamic flow scheduling outperforms the static schedulers. It also points that, central scheduler with global knowledge is key point for network performance.

B. Datacenter Traffic Control: Understanding Techniques and Trade-offs

This paper is a datacenter network review for different network architectures, topologies, traffic properties, traffic control challenges and objectives. Authors discussed datacenter network management schemes, transmission control, load balancing, multi-pathing and scheduling techniques in terms of performance, cost and complexity.

Nowadays datacenters require high bandwidth network connections. Modern systems like cloud computing, web, video delivery or social networks and users who use these systems demand faster delivery. The datacenter network demand is growing very fast and this requires high bandwidth in the network. However, DC networks can fail and this makes networks asymmetric and decreases the bisection bandwidth. There are different algorithms which can solve these problems and this paper reviews each of them.

The first topic is comparing 7 different network topologies and explaining their benefits. Fat-Tree topology uses aggregate layer between the edge and the core layers. This benefits a high number of hosts with low-cost switches. Leaf-Spine is good for expanding and load-balancers but scaling it is more expensive. VL2 is similar to Fat-Tree but it requires less cables for aggregation to core switches. JellyFish topology uses random settings. It is more expandable,

supports more servers and has higher throughput in failed state than Fat-Tree. DCell is the easiest to expand and doesn't have single point of failure; however, it requires custom routing algorithm. BCube can work in hardware or software but software one uses lot of processing. Among the other algorithms, BCube is the most flexible one to switch failures. The last topology is Xpander and its very similar to JellyFish. The difference between them is connection rules.

The next section is network traffic properties and properties like packet sizes, flow distributions and arrival times change with running application. For example, web search flows are shorter and smaller than the computing tasks.

There are 7 particular traffic control challenges over DC networks. Authors define them as follow. Unpredictable Traffic Matrix makes capacity planning and traffic engineering (TE) harder. Mix of Flow Types and Sizes makes flow scheduling more difficult. Traffic Burstiness is bad for switches and network performance. Packet Reordering requires high CPU utilization which causes latency. Performance Isolation hardens network attack detection. The Incast Problem needs larger switch buffers and better traffic control. The last one is The Outcast Problem and like The Incast Problem it requires advance traffic control.

Traffic control objectives vary from operators to tenants to end-users. Operators wants higher utilizations so that they can rent their system to more tenants. Tenants want fair share of resources and end-users would like to have faster services. Thus datacenters have to minimize flow completion times for faster responses and minimize missing deadlines numbers for quality of service requirements. They also must maximize the utilization to increase the DC's revenue. Moreover, there should be fairness in resource allocation in order to meet service level agreements (SLA). The last thing operators consider is energy consumption and efficiency of the DC. They deal with it because of the energy costs.

Distributed, centralized and hybrid are the 3 approaches that are used in datacenter traffic control management. Distributed method is more reliable and scalable. It can also use feedbacks to improve performance. However, network wide management is harder than the centralized one. On the other hand, centralized can access global view of the network and all the flow features. Management and applying network-wide policies are easier in centralized systems. Disadvantages of centralize method are failures, hacker attacks and latency for collecting all the data from the network. Lastly, hybrid method uses both central controller and distributed controllers. That's way its reliable and scalable like distributed and has higher performance like centralized. The con of the hybrid system is complexity. Also, it doesn't perform as good as centralized one but, it has better tolerance to failures than it.

Datacenter traffic control techniques consist of Transmission Control, Traffic Shaping, Prioritization, Load Balancing, Multipathing and Scheduling.

Transmission Control manages the data flow which transmitted to network. Changing window size of the connection is one of the methods and its called window-based technique. For instance, MCP algorithm can change windows size according to switches' queue length. There are also different approaches like rate-based and token-based transmission controls.

Traffic Shaping is making sure that performance of the network meets the necessary standards. It is very useful in DCs with many users. There are 2 methods in Traffic Shaping: Rate Limiting and Packet Pacing. Rate Limiting restricts the leaving packets' transmission rate. It can be used

with hardware or software. Hardware Rate Limiting performs better but costs more than the software one. Packet Pacing equals time between each packet with adding extra time between them. It decreases burstiness and latency. Like Rate Limiting, it can be used on hardware or software and its accuracy depends on it.

The third method is prioritization. Giving different priorities to flows can benefit the overall network performance. Prioritization has 5 features: classification (dynamic/static), criteria (flow size/deadline/class), location (switch/end-host), implementation (layer/custom) and enforcement (strict/non-strict). Flow priorities can use flows' characteristics such as deadline or size. Also, system can check these priorities in switch level or in end-host level. An example of dynamic prioritization is Dynamic Packet Prioritization (DCP), it uses 2 queues. One of the queues is for higher priority flows and the other one is for normal flows.

Load Balancing is another method for traffic control. It aims to reduce the number of highly utilized links and distribute the load across all connections. So it tries to boost overall network utilization. Load Balancing can be dynamic or static. ECMP uses hashing for routing and flows' path cannot change so, it is a static load balancer algorithm. On the other hand, Hedera [1] is a dynamic load balancer so flows' path can change in transmission. Hedera doesn't calculate dynamic paths for each flow. It waits for bigger flows and reschedules them for higher bisection bandwidth. Load Balancing algorithms can vary for their calculation rate. They can be per flow, per packet, per flowlet or per flowcell base. For instance, both ECMP and Hedera are per flow load balancers but Valiant Load Balancing (VLB) is per packet load balancer. Per packet one requires packet reordering so performance may be an issue. Also, per flowlet and per flowcell use group of packets. The size of the per flowlet is dynamically changing, while the size of the per flowcell is initially constant.

Multipathing is basically splitting flows into sub-flows and transmitting each sub-flow through distant path. Multipathing increases network performance and utilization other than under heavy workloads.

The last traffic control form is scheduling. Scheduler's goal is to increase performance/utilization while decreasing the latency. Schedulers could use features like reservation, redundancy, deadline-awareness, disciplines, pre-emption, jittering and ACK knowledge. However, each has drawbacks. For instance, reservation increases latency and it is hard to schedule network-wide. Redundancy is not very effective under heavy workload. Deadline-awareness requires extra calculation for deadline estimations. Disciplines are good for only one performance metric. Pre-emption isn't good for similar flows. Also, jittering and ACK control isn't very effective.

To sum up, this paper summarizes datacenter network problems, challenges and their solutions with their trade-offs. Authors consider datacenter operators, tenants and end-users while reviewing each aspect of the datacenter network. The paper also shows that traffic control and management in DC network is necessary and there are tons of different ways for it. However, each method has their own strength and drawback about performance of the network.

C. Efficient Load Balancing Over Asymmetric Datacenter Topologies

Load balancing in data center networks is necessity for high bisection bandwidth. Packet spraying method works well under symmetric networks but it cannot perform well under asymmetric topologies. This paper introduces Symmetric Adaptive Packet Spraying (SAPS) [3] for asymmetric network load balancing with symmetric virtual topologies (SVT).

Most of the datacenter services use by users and they expect minimal response times. Since data center network requires higher performance, most of the DCs use multi-rooted tree topology. Every host has multiple paths between every other host and these multiple paths requires load balancing for efficient usage. Static load balancers such as Equal Cost Multi-Path (ECMP) aren't as good as the dynamic ones like Packet Spraying (PS). PS performs near-optimal over symmetric topologies. Nonetheless, network failures broke symmetry and they become asymmetric. Failures are common among DC and PS cannot achieve good results over asymmetric networks. Although there are some asymmetric topology load balancers, they cannot operate near optimal results. That is why authors want to work on near-optimal performance PS over asymmetric topologies. They tested SAPS and similar algorithms on realistic DC network traffic.

There are 2 types of network failures: partial failures and full failures. Partial failures are bandwidth restrictions and they are common in the DC network. For example, in 10 Gbps connection, using 1 Gbps bandwidth is this kind of failure. In full failure, the link between hosts become completely unusable. Both kind of failures creates asymmetric network paths in the DC network tree.

SAPS have 3 goals: high throughput, robust to asymmetry and compatible with OpenFlow switches. Since PS uses per-packet load balancing, SAPS uses the same method. Yet, SAPS avoids using all the paths for spraying like PS. Due to the asymmetric network's links, using all paths performs sub-optimal. To overcome the asymmetry, SAPS uses symmetric virtual topology. SDN controller helps SAPS for installing flow table rules to different virtual topologies. Each flow uses single SVT. SVTs don't include any failed links due to asymmetry. Also, SAPS deals with full failed links with better queueing techniques. Authors used Data Center TCP (DCTCP) to reduce queue sizes. SAPS uses OpenFlow switches and controller module. The controller detects link failures and updates SAPS about it.

Authors used 2-tier tree with 4 leafs and 16 hosts for testing the algorithm. All links are capped to 10 Gbps and they used real-world DC workloads such as web search and data mining. In partial failure state with data mining workload, SAPS performs 20-50% better than the ECMP and 25-49% better than the PS. Also in web search workload SAPS outperforms ECMP by 57-93% and PS by 86-98%. Then they tested on full link failures, SAPS were ~7% better than the ECMP but its performance was similar to PS.

In the future, the authors want to add energy consumption aspect into the algorithm. In conclusion, this paper presents SAPS (Symmetric Adaptive Packet Spraying) load balancing algorithm for asymmetric network topologies in data centers. SAPS uses Packet Spraying method with Symmetric Virtual Topologies (SVT) to achieve better performance in the case of failures.

D. Sorted-GFF: An Efficient Large Flows Placing Mechanism in Software Defined Network Datacenter

Sorted-GFF (Global First Fit) [4] is a flow scheduler for elephant flows (large size flows). This algorithm works with Software-Defined Networks (SDN). Most of the data centers use Fat-Tree networks with core and aggregate switches. There are similar algorithms for multi-path routing. However, since data in the DC getting bigger each year, Sorted-GFF focuses on the elephant flows.

Equal Cost Multi-Path (ECMP) is using hashes for routing and it is a static flow scheduler. Since it is static, it cannot apply different methods on large (elephant) and small (mice) flows. Consequently, mice flows wait for elephant flows to finish. Thus, elephant flows have to be handled efficiently and differently from mice flows.

Hedera [1], Mahout [5], DevoFlow [6] and LABERIO [7] are the similar algorithms but each of them have their own limitations. For instance, Hedera has late congestion detection and fixed thresholds while Mahout lacks of changes/failures and it requires host modification. In addition, DevoFlow requires switch modification and LABERIO needs host modification. The author focuses on Hedera and try to improve its performance and simplicity. Improving the efficiency and dynamic/fast responses to the network updates are the 2 main goals of the paper. Furthermore, for simplicity and scalability needs, it is preferable not to use extra tools or modifications.

Sorted-GFF tries to improve Hedera so, it uses ECMP for routing the flows which is under the threshold (10% of the total bandwidth) value. ECMP can assign 2 or more elephant flows to the same path. In that case, adaptive rerouting is needed.

Hedera uses natural demand estimation algorithm for detecting the elephant flows. If the natural demand of the flow is higher than the 10% of the total link bandwidth, Hedera acts like it is an elephant flow. However, the author believes that 10% of the bandwidth is not large enough to congest the link. Because 10% threshold means there is still available 90% bandwidth. Therefore, Sorted-GFF uses higher threshold value than the Hedera. Nevertheless, Sorted-GFF doesn't change Hedera's natural demand estimation algorithm, it just adjusts the threshold value.

Hedera uses GFF for rerouting/finding new path. Because it is simple and efficient algorithm for that task. Nonetheless, GFF cannot guarantee that it will find a path for all the flows. Sorted-GFF tries to solve this problem while not increasing the GFF's complexity. Like GFF, Sorted-GFF uses estimated natural demands thus, it doesn't require extra monitoring or processes.

They tested ECMP, GFF and Sorted-GFF algorithms on simulation. The simulation layout is similar to Hedera's [1] test bench (16 hosts and 20 switches). The difference between Hedera's test and Sorted-GFF's simulation is SDN. Source and destination host selection methods are the same as Hedera (Stride, Staggered Prob., Random). As a result, Sorted-GFF performs ~5% better than the GFF for rerouting.

Finally, this paper presents Sorted-GFF algorithm for data center network flow scheduling. It uses Hedera as a base and tries to improve its GFF algorithm with Sorted-GFF. Also, author believes that 10% threshold is not enough for elephant flow detection. Lastly, this paper

compares GFF and Sorted-GFF algorithms but Hedera also uses Simulated Annealing (SA) algorithm for the same job. However, this paper's evaluation part doesn't include SA performance results.

E. JITeR: Just-In-Time Application-Layer Routing

JITeR [8] (Just-In-Time Routing) is a reliable and in time routing algorithm for geo-distributed information infrastructures (GDII) and clouds. JITeR provides messages' latency and reliability in wide-area networks. It uses overlay networking and multihoming techniques for messages. Authors tested this algorithm in simulations and in Amazon EC2 clouds.

Paper mentions that most of the network messages/packages have deadlines and some of them have to strictly meet these deadlines. However, due to network faults like congestion and omission, some of the deadlines are missed. JITeR tries to solve this missed deadline problem while considering the potential faults in GDIIs. It also considers power outages, cyber attacks like DDoS attacks. Message routes that selected by JITeR algorithm meets deadlines and they don't block the network traffic. It uses node set and finds a path between these nodes. These nodes represent geo-distributed datacenters and between each node, there is an ISP (Internet Service Provider). Since nodes are distributed worldwide, the ISPs between nodes are different from each other.

JITeR tries to accomplish timely and reliable communication. It has 3 objectives to work with current GDIIs. These are compatibility with current GDIIs, no changes in wide-area IP network and the cost. Authors aim for practical solution for current systems with low-cost.

JITeR messages' deadlines have high priority. That's why it sends each message through more than one overlay channels. It uses one base channel and backup channels. Since reliability is important for JITeR, it uses different ISP channels for each message. If the source node doesn't get the acknowledgement signal, the message would be retransmitted. JITeR algorithm selects these channels for messages while considering each messages' deadline. Therefore, some of these channels are not the fastest way but they are more reliable and they meet the message's deadline.

There are similar algorithms that do the same job as JITeR. We need to understand how these algorithms work in order to compare the performance of the algorithms. One of the similar one is Best-Path. It sends the message through the best overlay channel and retransmits it at most 3 times. Another one is Flooding method which sends the message through all overlay channels at the same time. Multi-Path algorithm uses 2 overlay channels which one of them is direct channel and the other one is randomly selected. Hybrid technique sends message through 1 direct channel. If the message delivery fails, it uses 4 random overlay channels for retransmit. Round-Robin selects one channel and changes it in circular format. It also retransmits the message at most 3 times. The final approach is Primary-Backup. It uses 1 channel until it fails and in case of failure it uses another channel. There are also 2 versions of JITeR algorithm. These are JITeR^0 and JITeR^1. JITeR^0 doesn't use any backup channel but JITeR^1 uses 1 backup channel other than the main one. Also, all of these algorithm use overlay and/or multihoming methods.

Since GDIIs are using WAN-of-LAN structure, JITeR designed for these kind of structures. In JITeR architecture WAN is the inter-node communication part and LAN is the intra-node communication element. All nodes have matrix that holds every other nodes information. Thus each node knows all the other nodes in the network. This method provides better routing and monitoring. When a node fails or updates, JITeR updates all other nodes' matrixes so that the network can stay up-to-date. JITeR algorithm finds a path that has at most one-hop. That means, messages have to cross at most 1 other node before arriving to the destination node. JITeR retransmits messages in case of failure. Retransmission stops with ACK signal or reception deadline. JITeR also uses backup channels. It selects backup channels that have least interaction with base channel and that can deliver the message in time.

They tested all algorithms on Italian power grid GDII simulation. They used 3 different fault scenarios. These scenarios are fault-free (FF), accidental faults (AF) and crisis (C). AF represents shorter failures (%30 30 seconds, rest shorter), while C represents longer ones (%80 30+ seconds, rest shorter). Authors used 2 different aspects for comparing the algorithms: missed deadlines and % of extra messages sent. In FF state, none of the algorithms missed any deadline and only JITeR^1 (100), Flooding (2410) and M.P. (100) sent extra messages. In AF state, only JITeR^0, JITeR^1 and Flooding didn't miss deadlines. And extra message numbers were similar to the FF state. Most of algorithms' extra % is between 0-1 but, JITeR^1 (100), Flooding (2410) and M.P. (100) are higher than it. Finally, in C state all of the algorithms have missed deadlines. The best ones were JITeR^0 (97), JITeR^1 (37) and Flooding (5) while the worst one was R.R. (9600). Extra messages % was higher for all of the algorithms but again Flooding (2410) was the first one. The others were B.P. (1.76), JITeR^0 (49.29), JITeR^1 (167.63), M.P. (100), Hybrid (26.62), R.R. (18.67) and P.B. (1.78).

The simulation results show that Flooding is best for deadlines however, it uses too much extra messages which can slow the network. The second best for the missed deadline ratio is JITeR algorithms. Both of them have lower missed deadline numbers than the other algorithms. JITeR^1 has more extra message percentage than the JITeR^0 but its deadline count is bit lower. For all 3 fault state JITeR algorithms have the best overall performance for both message deadlines and number of extra messages.

Lastly, JITeR is one of the best timely and reliable message routing algorithm for GDIIs. While delivering the messages on time, it doesn't affect the network's overall traffic/performance.

F. VARMAN: Multi-Plane Security Framework for Software Defined Networks

VARMAN (ad**V**anced multi-pl**A**ne secu**R**ity fra**M**ework for softw**A**re defined **N**etworks) [9] is a Network Security and Intrusion Detection System (NIDS) framework for Software Defined Networks (SDN). SDN has its own "anti-DDoS/botnet" security but its not sufficient for DC network security. NIDS has Machine Learning (ML) solutions for better security; however, these ML-based methods require high processing power and memory space. VARMAN tries to improve NIDS security without requiring too much processing power and memory also, without increasing the network latency.

DC network security is an important topic. SDN architecture makes network management and security easier. NIDS and ML algorithms provides more secure networks for SDN; however, they require high computational resources. So, these methods aren't favorable for real-time systems. VARMAN provides secure network without using too much resources. It has to solve 3 sub-problems: "(a) anomaly detection in data-plane, (b) classification in control-plane and (c) collaborative interface between the layers and downstream services".

ML-based security algorithms can use either shallow ML or deep learning. Both of them have their pros and cons. But, there is also hybrid approach for better accuracy with less computational recourses. VARMAN uses hybrid approach at the control plane for network attack detection. It combines shallow and deep learning ML techniques with a non-symmetric stacked autoencoder and an improvised random forest classifier. Also, it provides efficient reduction system such as feature selection and data filtering for decreasing the overall processing. It also benefits multi-plane SDN stack (Data plane, control plane, application plane) to boost the network security.

VARMAN requires switch modification for better network monitoring and anomaly detection. When one of the switches detects an attack, it alerts the controller unit and passes the attack data. Controller classifies the attack with ML-based algorithm. Then, controller coordinates all attacked switches for classified attack's defense-action. That means, controller operates the data plane/switches; nonetheless, every switch has embedded flow monitor module for attack detection. Cross-coordinating layer collects data from all of the layers and store these data in database for network topology map. There is also an Anti-Spoofing module for avoiding miscalculated detection.

VARMAN's ML workflow consists of preprocessing, normalization, feature selection and reduction and classification. For classifying the attack type, authors used features like length of connection, protocol type, max expire time, packet per second (PPS), …

For testing the accuracy and the performance of the algorithm, authors created a simulation with large-scale DDoS/botnet attack scenarios. Attacks are divided into 2 as flooding (constant and increasing) and slow-rate attacks. VARMAN's detection rate was 96.8% for slow-rate and 100% for flooding attacks. They also tested different number of features for detection. The best accuracy obtained with 10-12 features. More than 16 features drop the accuracy of the results.

VARMAN has different defence mechanisms for each attack type. For instance, DoS attacks detected at data plane and defence method for DoS is block/drop. However, application layer attacks are harder to detect thus they detected in control plane. VARMAN's protection method for app layer attacks is block/drop/remediate.

Another test scenario compares normal state, no defence scheme, CSD-OVS and VARMAN's CPU utilization in attack. In normal state workload wasn't affected. In no scheme state, attacked switches' controller's utilization was 95%. Using CDS-OVS reduces CPU utilization from 95% to 65% but flows encounter some additional latency. With VARMAN, utilization dropped to 65% but latency wasn't an issue. Also, VARMAN didn't affect other controllers' utilization.

The final test was about ML accuracy and performance. Authors compare VARMAN's algorithm with Deep Belief Network (DBN). They used real network data with attacks and DBN's average accuracy was 97.1% while VARMAN's accuracy was 99.5%. Also VARMAN's precision and recall values are higher than the DBN besides, false-positive and

false-negative values are lower. These results aren't that superior but VARMAN's main goal is using fewer resources. Therefore, they tested both algorithms' training time and required memory space. DBN's training time was 23422 seconds and it requires 268542 KB memory. VARMAN's training time was 1202 seconds and it only requires 24568 KB memory. These results show that VARMAN and hybrid approach both increase the accuracy and reduce the computational resource need.

To conclude, this paper introduces VARMAN framework for DC networks with SDN. VARMAN benefits multi-plane SDN, hybrid ML algorithm and it uses fewer resources. It detects and classifies diverse attack types and prepares defence-actions against each type of attack. While doing these, it doesn't use too much resources and it doesn't add extra latency into the network flows.

IV. Comprehensive Summary

My topic for this literature review was datacenter networking. I've read 6 different papers about DC networks. These 6 papers' subject distribution is like that: 3 load balancing/flow scheduling algorithms (Hedera [1], SAPS [3], Sorted-GFF [4]), 1 inter-datacenter routing algorithm (JITeR [8]), 1 network security algorithm (VARMAN [9]) and 1 traffic control algorithms review [2].

While I was reading these paper, I noticed that Software-Defined Network (SDN) is very popular in the networks, especially in the datacenter networks. SDN makes both network management and monitoring easier and most of the algorithms (like Sorted-GFF and VARMAN) built on top of SDN architecture.

3 of the papers/algorithms tries to solve the load balancing issue in the DC networks. It's one of the most popular issues in DC networking and there are lots of different algorithms for solving this problem. The authors of the 3 papers agree on that dynamic load balancers are superior over static ones. That's way all of them compares their algorithm performance with most popular static load balancer Equal Cost Multi-Path (ECMP). Although, Hedera uses static hash mapping before threshold, it reroutes the flows when they exceed the threshold value. Thus, it is a dynamic load balancer and it's the oldest algorithm (2010) among others. Hedera algorithm's base is decent for DC networks. Since its one of the earliest papers in load balancing subject, most of the following papers use its techniques or compare their algorithm with Hedera. For this exact reason, Sorted-GFF tries to improve it with SDN and better placement algorithm. The downside of the Sorted-GFF paper is not comparing Sorted-GFF algorithm with Simulated Annealing (SA) algorithm. SA has better results than GFF in Hedera paper [1] but, in Sorted-GFF paper, author only compares GFF and Sorted-GFF algorithms performance. The third and the last paper (SAPS) tries to improve load balancers performance in case of network failures. It uses Packet Spraying (PS) algorithm and improves its performance in asymmetric topologies which caused by failures.

At first JITeR [8] algorithm might look like a load balancing algorithm for datacenters. Nonetheless, its an immediate and trustworthy message routing algorithm for inter-datacenter connections. It tries to improve connection performance and fault tolerance between geo-

distributed information infrastructures (GDII). Since most of the datacenters/clouds/servers have a distributed center around the world, this kind of algorithms are important for the DCs.

Another critical subject in DC network is security. VARMAN makes DC networks more secure against attacks like DDoS or botnet. It uses machine learning techniques to detect network attacks. There are similar algorithms which do the same job with ML. VARMAN's difference is using hybrid method. Hybrid means using both shallow and deep learning algorithms to increase accuracy and performance of the ML. Lastly, ML is one of the most popular topics in computer science and in the future, there will be numerous applications/uses of it in the DCs.

The last paper [2] is a complete review of the DC network traffic control algorithms. It explains most of the problems, challenges and their solutions in DC networks. It compares various network architectures and explains their limitations. Then, it continues with traffic control objectives and challenges. It clarifies each objective and what are the challenges to achieve them. Lastly, it analyzes traffic control algorithms that solve these challenges and achieve the given objectives. This paper is excellent for beginners who want to understand the datacenter networking.

To summarize, datacenter network is a huge research area. Even though datacenters have different research topics like energy consumption, DC network's most popular subjects are performance, reliability and security. In this project, I've reviewed 6 papers that focuses on these 3 arguments.

V. References

[1]. M. Al-Fares, *et al.*, "Hedera: Dynamic Flow Scheduling for Data Center Networks", *Nsdi*. Vol. 10. No. 2010. 2010. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.212.9046. [Accessed Dec. 19, 2019].

[2]. Mohammad Noormohammadpour and Cauligi Raghavendra, "Datacenter Traffic Control: Understanding Techniques and Trade-offs", *Communications Surveys and Tutorials, IEEE Communications Society, Institute of Electrical and Electronics Engineers*, Vol. 20, No. 2, pp. 1492-1525, 2018. [Online]. Available: ⟨10.1109/COMST.2017.2782753⟩. ⟨hal-01811647⟩. [Accessed Dec. 19, 2019].

[3]. S. M. Irteza, *et al.*, "Efficient load balancing over asymmetric datacenter topologies", *Computer Communications*, Vol. 127, pp. 1-12, Sep 2018. [Online]. Available: https://journals.scholarsportal.info/details/01403664/v127icomplete/1_elboadt.xml. [Accessed Dec. 19, 2019].

[4]. J. A. Rashid, "Sorted-GFF: An efficient large flows placing mechanism in software defined network datacenter", *Karbala International Journal of Modern Science*, Vol. 4, Issue 3, pp. 313-331, Sep 2018. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S2405609X18300368. [Accessed Dec. 19, 2019].

[5]. A. R. Curtis, *et al.*, “Mahout: low-overhead datacenter traffic management using end-host-based elephant detection”, *in: INFOCOM*, 2011 Proceedings IEEE, IEEE, 2011.

[6]. A. R. Curtis, *et al.*, “DevoFlow: scaling flow management for high-performance networks”, *ACM SIGCOMM Computer Communications Rev. 41*, Vol. 4, pp. 254-265, 2011.

[7]. H. Long, *et al.*, “LABERIO: dynamic load-balanced routing in OpenFlow-enabled networks, in: Advanced Information Networking and Applications (AINA)”, *2013 IEEE 27th International Conference* on, IEEE, 2013.

[8]. A. Bessani, *et al.*, "JITeR: Just-in-time application-layer routing", *Computer Networks*, Issue 104, pp. 122-136, May 2016. [Online]. Available: https://orbilu.uni.lu/handle/10993/28232. [Accessed Dec. 19, 2019].

[9]. P. Krishnan, *et al.*, "VARMAN: Multi-plane security framework for software defined networks", *Computer Communications*, Vol. 148, pp. 215-239, Dec 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419308217. [Accessed Dec. 19, 2019].

Baran Kaya, 400284996, kayab@mcmaster.ca