Cureus Journal of Computer Science

Open Access Original Article

Dynamic Bandwidth-Aware Link Aggregation Control Algorithm for Self-Hosted Learning Management Systems Data Center

Received 03/26/2025
Review began 03/30/2025
Review ended 07/03/2025
Published 07/14/2025

DOI: https://doi.org/10.7759/s44389-025-04070-w

Bright K. Kunu [1] , Joshua C. Dagadu [2] , Stephen K. Pometsey [2] , Oliver K. Boansi [2]

1. Information Technology, Ghana Communication Technology University, Accra, GHA 2. Information Technology, Akenten Appiah-Menka University of Skills Training and Entrepreneurial Development, Kumasi, GHA

Corresponding author: Bright K. Kunu, brightkunu@gmail.com

Abstract

Online learning management systems (LMS) rely heavily on consistent network connectivity to meet the ever-changing demands for its resources. Ensuring the availability of sufficient bandwidth is particularly important but challenging for self-hosted LMS, which is characterized by varying bandwidth requirement. Many self-hosted systems are fraught with low bandwidth and hence deliver poor-quality service to end users. The various interventions for addressing bandwidth wear-outs in high traffic networks are rather expensive, and inefficient for self-hosted systems as they result in link redundancy. Therefore, this study proposes a framework that uses software-defined networking (SDN)-based Link Aggregation Control Protocol (LACP) to allocate network bandwidth according to the dynamic traffic demands of the LMS application. The dynamic bandwidth-aware LACP algorithm monitors and analyzes traffic on the LMS at regular time intervals and assigns network bandwidth in response, accommodating the bandwidth needed for the specific instance. The system is simulated for a two-LAG and four-LAG setups using the SDN Ryu controller. The result of the simulation shows an efficient bandwidth assignment according to the dynamic traffic requirements of the LMS. In all, an average bandwidth of 11,900.4 Mbps was recorded for all user datagram protocol streams, amounting to about 60.5% improvement in bandwidth reservation compared to 7,413.33 Mbps recorded in a static LACP implementation. Also, the system demonstrates a quick recovery by adjusting and re-assigning link bandwidth when one or two links are down. However, the result also shows a weakness of the Ryu controller to efficiently assign and utilize bandwidth as the number of aggregated links increases, resulting in increased packet retransmission rates. This suggests that the number of links aggregated into a LAG should be carefully selected according to the SDN controller's ability to fully utilize them.

Categories: Networking Technologies, Cloud Networking, Soft Computing
Keywords: software-defined networking (sdn), data centre, bandwidth awareness, link aggregation control protocol (lacp), self-hosting

Introduction

The goal of every network administrator is to correctly map the high-level intent (policy) to the low-level forwarding behavior of the network [1]. Several interventions have been introduced in the networking sphere to efficiently map the high-level network policies to their traffic forwarding behavior. These include network bandwidth availability [2], traffic engineering [3], resilience against failure [4], scalability, decentralized visibility of hardware [5], and interoperability [6].

These interventions have been slow in meeting the ever-increasing requirements of recent network systems. In recent years, software-defined networking (SDN) techniques are being adopted to virtualize network systems. Studies have shown that the SDN is a suitable and efficient virtualization technique for both guided and unguided networks due to its superior architecture in network slicing, logical centralized control and customization, and network programmability [7,8]. Furthermore, the SDN's superior structural architecture makes it well suited to provide an efficient backhaul network virtualization for mobile networks [9].

Following the advancements in SDN implementation technologies and standards for network component virtualization, link aggregation (LA) mechanisms for communication link virtualization have been introduced. Other SDN-based network component virtualization standards include the FlowVisor for switch virtualization and PFlow for service virtualization [10,11]. LA focuses on increasing the transmission capacity of network links by bundling two or more Ethernet channels into a single logical link called the link aggregation group (LAG). LAG is commonly used to provide aggregated link throughput and quick fault tolerance in the event of a failure in one or more physical links [12,13]. These link shaping capabilities makes the SDN-based link aggregation control protocol (LACP) implementation a more suitable link virtualization approach for self-hosting learning management systems (LMS).

How to cite this article
Kunu B K, Dagadu J C, Pometsey S K, et al. (July 14, 2025) Dynamic Bandwidth-Aware Link Aggregation Control Algorithm for Self-Hosted Learning Management Systems Data Center. Cureus J Comput Sci 2 : es44389-025-04070-w. DOI https://doi.org/10.7759/s44389-025-04070-w

According to Shurygin et al. [14], an LMS is a software application used for the administration, documentation, tracking, reporting, automation, and delivery of educational courses, training, or learning and development programs. It serves as a platform for online or virtual learning and the sharing of educational resources. In recent years, many open-source LMSs have become popular due to their pervasive use in education delivery and management. However, studies have shown that many instances of self-hosting the LMS are associated with notable technological challenges, which severely affect its accessibility and users' experiences across the globe. For instance, Almaiah and Al Mulhem [15] assert that lack of appropriate technological infrastructure is a major setback for LMS accessibility and use in some Saudi Arabian universities. One of the bases for this study is the assertion of Qu et al. [16] that the networking infrastructural inefficiencies are deepened when allocating multimedia resources on the LMS. These resources allocation challenges are common drawbacks for operating on-premises or self-hosted LMS infrastructure, which is usually deepened when the system usage scales up [17].

Cloud hosting holds great benefits to mitigating the technical implementation challenges with the LMS infrastructure [18,19]. Such challenges as cost, convenience, scalability, and efficiency are proficiently catered for by outsourcing the LMS hosting to a cloud provider. However, this comes at the back of privacy, data ownership, and centralized point of failure to the universities [17,20]. Therefore, Abdulmohson et al. [17] concluded that opting to self-host the LMS was the best choice for Kufa University since it had its own on-premises infrastructure, and the value of self-control cannot be traded for the offerings of cloud hosting. Self-hosting offers an opportunity for organizations and individuals to own their digital infrastructure by taking control of managing its efficient running and security needs [20]. A layered view of SDN architecture is given in Figure *1*.

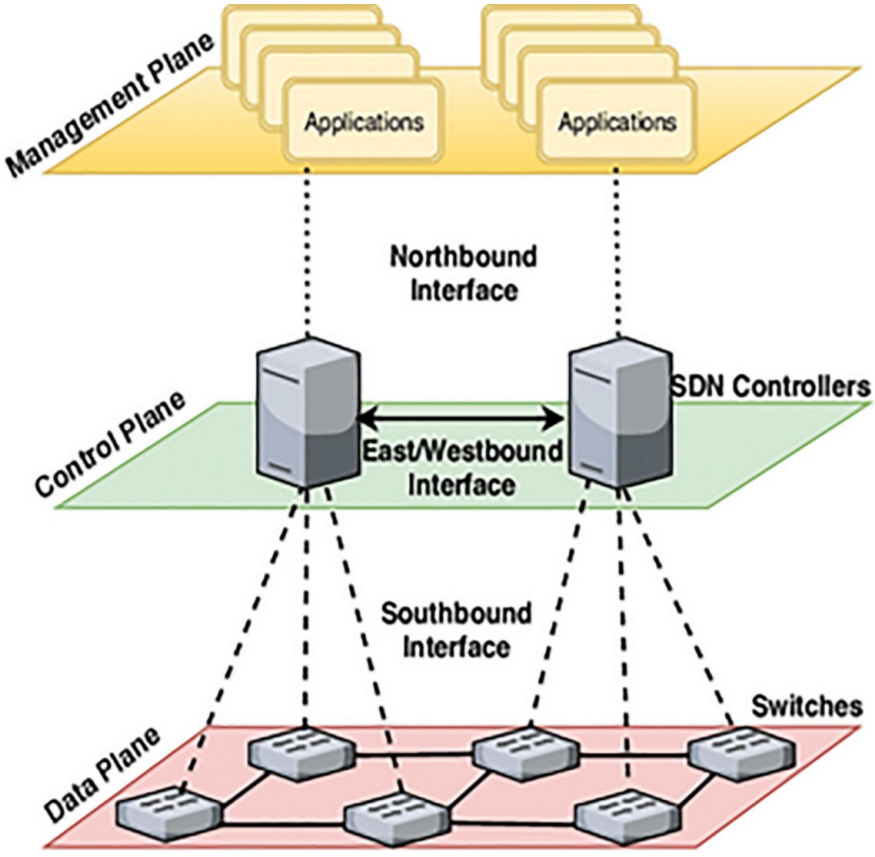

FIGURE 1: Layered view of SDN architecture

Source: Adapted from Latif et al. [21]

SDN, Software-Defined Networking

Adopting SDN based LACP for the self-hosted LMS infrastructure will help the universities to close in on the infrastructural benefits cloud hosting offers. This is because SDN supports network slicing and the creation of virtual application tenants and at the same time serves as a platform for LACP implementation for virtual bundling of the communication links [22]. More importantly, the controller-based architecture of the SDN makes it possible to programmatically optimize the virtualization and utilization of all

network components [23]. One advantage the SDN architecture brings is that it does not rely on complex and very expensive networking devices to run [24]. Rather, it makes it easy to cut down cost by maximizing existing on-premises infrastructure while simplifying construction and maintenance. However, there is a lack of studies on SDN-based LACP implementation, which performs a dynamic scaling of the network link bandwidth to support the varying traffic needs of the online application.

This paper proposes a dynamic bandwidth-aware LACP algorithm for self-hosted LMS. The proposed algorithm specifically addresses the dynamic link bandwidth scaling of the self-hosted educational system in response to the traffic requirements, while making room for failover in moments of link failure. The proposed system is modeled to estimate and respond to three main traffic scenarios (low, medium, or high). The algorithm is simulated in Mininet for bandwidth reservation and QoS of the self-hosted LMS, according to the dynamic application contexts and the bandwidth requirements of the network traffic, using SDN OpenFlow Ryu controller. The Ryu controller helps to reduce the complexity of the high-level intents of the educational system into low-level IT networking policies, which is implemented using a lightweight algorithm.

Materials And Methods

Preliminaries

SDN

The SDN architecture splits the entire network stack into control and data forwarding planes and slices the underlying network to ease construction, forwarding, and management [1]. The data forwarding plane consists of the traffic forwarding devices such as switches and routers [25]. Also, the control plane consists of the network hypervisor, and the application control plane [26]. These network slicing functionalities of the SDN enables network engineers and administrators to compose software programs to manipulate the physical and logical map of each network slice [27-29].

LACP

Link aggregation is a method for bundling two or more individual ethernet links together so that they can function as a single logical link [30]. The bundled ethernet channels are called LAG; the protocol that runs and monitors the LA is LACP [30]. Traditionally, LACP is configured to enhance the handshake of the LAG members and define a load-balancing technique for communication between the link aggregated devices [31]. Advancements in LACP use cases range from a simple bandwidth improvement [13], to the selection of a suitable SDN controller [12], and network traffic engineering [21].

Proposed system

The dynamic bandwidth-aware system is programmed to ease congestions on the LMS by automatically adjusting the campus network bandwidth to improve end-users experience on the LMS. The system performs bandwidth reservation according to the data flow context of the LMS. It has discrete events that monitors packet flow characteristics of the LMS, define rules to interpret the flow characteristics, and perform customized actions on the packets.

The proposed system first observes and generates a log of data flow characteristics on the LMS data center. These include port negotiation information, port status information, and bandwidth requirements calculation. If a link is down or active, its port status is captured accordingly for the negotiator and its bandwidth accordingly recorded. The system then triggers the traffic monitoring event which gives its context inference. Thus, it defines heavy or low traffic situations. Low traffic flow context is set to bandwidth requirements below 80% of available link bandwidth. In this context, the system basically transmits packets over the available individual physical links since less network resources will be used in the process. The rule for heavy traffic bandwidth requirement is set to 80% of the requested and available link bandwidth. Based on the recorded flow context inference, the system will calculate its bandwidth requirement and make same available to enhance unhindered traffic flow. In this case, the controller will add links to LAG and transmit the traffic over the LACP links.

The self-hosted LMS server is expected to exhibit a peak time behavior all day long. Thousands of remote concurrent negotiations for network resources must be processed throughout the day. Therefore, the proposed algorithm provides a traffic shaping mechanism to ensure all users have an overall favorable connection to their learning resources. For example, on a typical weekday, thousands of users compete for network access to real-time services from the LMS. The Ryu controller keeps a log of all network resource negotiations and estimate bandwidth requirements for concurrent port connection requests it receives.

Based on the estimated bandwidth requirement, the controller will sanction traffic flow over the LAG link or the physical links. This is done by calculating the bandwidth requirements of each arriving traffic and classifying it as low, moderate, or high traffic. Then, the system estimates the requisite link bandwidth

required to transmit the traffic using the formula:

$B(t) = minAi(t)$

where Ai is the available link bandwidth in the node i.

If the estimated traffic bandwidth B(t) at any time is less than the 80% thresholds for total negotiated link bandwidth, the SDN controller will transmit traffic over the available physical links. Thus, traffic with less than 80% of Ai will require less network resources to efficiently transmit traffic over the physical links.

When the estimated traffic bandwidth is greater than or equal to the 80% threshold for a negotiated link bandwidth, then the SDN controller will add links to LAG accordingly to increase the available link bandwidth to enhance efficient link usage and traffic flow. This is illustrated in the block diagram as shown in Figure *2*.

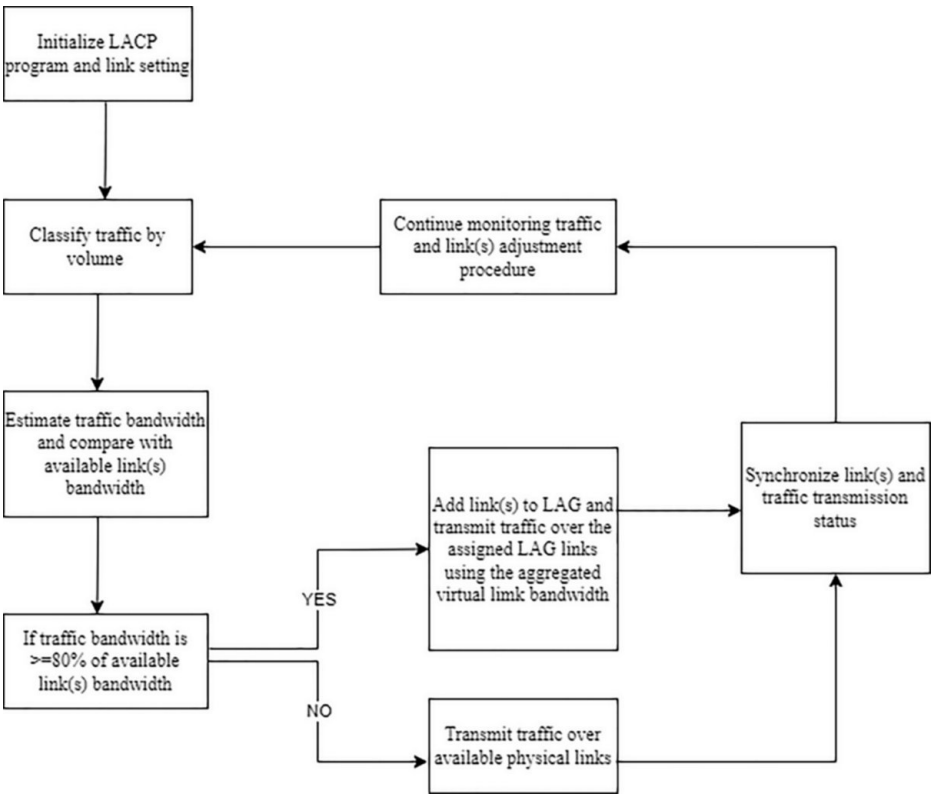

FIGURE 2: Block diagram for a bandwidth-aware LACP system

Source: Modeled by authors

LACP, Link Aggregation Control Protocol

Pseudocode Algorithm

```
InitializeLACP(links)

while (true)

 trafficVolume = classifyTraffic()

 trafficBandwidth = estimateBandwidth(trafficVolume)

 availableBandwidth = getAvailableBandwidth(links)

 if (trafficBandwidth >= 0.8 * availableBandwidth)
```

```
if (enoughLinksForAggregation(links))
syncLinksAndTraffic(links)
LAG = createLAG(links)
transmitOverLAG(LAG, trafficVolume)
else
handleInsufficientLinks()
else
transmitOverIndividualLinks(links, trafficVolume)
adjustLinks(links, trafficVolume)
wait for 10Sec (TrafficMonitoring)
end while
```

Context Awareness

The algorithm describes a python3 decorator @set_ev_cls in Ryu.controller.handler that monitors the event-port-modification class. This decorator performs the function of keeping port state information and calculating the throughput of requested links. In this study, the LACP system was configured to respond to links adjustment whenever the threshold of 80% weighted traffic bandwidth is reached for a requested link(s) at any time. At the interval of every 10 seconds, the system analyses the throughput in Mbps. Then, the system performs bandwidth calculation on each port of the simulated switch. The results of these calculations help the Ryu controller to make adaptations to the network and links. The main network adaptation provided for in this work is the decision to add more links to the initialized one or otherwise, and it is usually subject to calculated bandwidth requirement of traffic flow.

Dynamic Bandwidth Assurance

The system is configured to function in both active and passive LACP modes. LACP packets are transmitted in active mode whenever the learned ports recorded in the table-miss of the switching hub are matched and packets are forwarded to the learned port. This is done by binding the LAG group dpid to the learned ports in the switching hub, making the LAG group actively open for monitoring in the intervals of 10 seconds. The LACP settings (modeprobe 4) in the ubuntu system sets the logical EtherChannel (bond0) as the master channel and the physical EtherChannel h1-eth0 and h1-eth1 (for two-channel aggregation) or h1-eth0, h1-eth1, h1-eth2, and h1-eth3 (for four-channel aggregation) as slave channel depending on the number of links the system aggregates at any time. The system uses approximately 30 seconds to estimate bandwidth requirements and make appropriate adaptations. Whenever the 80% weighted bandwidth requirement is reached, a new physical link will be added to LAG, and traffic will be transmitted over them at the combined bandwidth of the logical interface created. Traffic that requires lower than the 80% bandwidth threshold of the requested link is transmitted using the default Open vSwitch switching mechanism. If the bandwidth requirement remains low for a continuous 5 minutes, the learned logical interfaces in the switching hub will be disabled. This demonstrates how the algorithm monitors the changes that occur in the slave state by the way events of the Ryu controller are executed.

In addition, if for any reason one of the links in the LAG fails or is disabled, the lacplib will direct traffic to the active link. So, in a LAG, the bonded physical channels always serve as active backups for each other. The actual bandwidth of the backup link remains the same as the bandwidth of the logical interface (bond0), but the maximum bandwidth that will be utilized will be the actual bandwidth of the backup link(s). For example, in the four-channel LAG, if h1-eth0 is down for any reason, the LACP data unit will be transmitted over the remaining channels (h1-eth1, h1-eth2, and h1-eth3), provided they are added to the LA Group. However, the actual bandwidth of the transmission is the summation of the weighted bandwidth of the used links.

The system monitors when a slave channel's current state orchestrates an idle timeout for the flow entry that performs packet-in of the LACP data unit. A timeout may occur because of a link failure or when the physical interface is disabled. The system will print the change of state to disable and then send a FlowRemove message to the switching hub. In this instance, packets-in flow is redirected to the Ryu controller, which decides to transmit the packet over a backup link. Therefore, when the enable/disable

state of a physical interface is changed, the FlowRemove event handler will delete all flow entries that use the physical interfaces included in the logical interface to which the disabled physical interface belongs. This makes LACP suitable for constructing highly available network links and provides a quick switch for fault tolerance.

Simulation

Simulation Materials and Tools

Firstly, Ubuntu server 21.04 was installed on a VMware Workstation 16 pro running on HP Pavilion laptop 15-cs1xxx, having 8 CPUs at a base clock rate of 1.60 GHz adjustable to 1.8 GHz, and 12,288 MB RAM. The VMware Workstation 16 pro virtual hardware specification includes 4 CPUs at a base clock rate of 1.60 GHz and 7,921 MB RAM.

OpenFlow1.3 was used for communication between the SDN control plane and Open vSwitch data plane in Mininet 3.0 environment. Mininet is a lightweight virtualization environment that uses lightweight Operating System containers (such as Linux shell) for building a complete network within a single OS instance. In addition, Mininet helps to verify the functional correctness of programming codes used in network experiments. Its components provide the basis for verifying performance properties for a much wider range of experiments. Scripts for the algorithm were written in Python3.

Traffic scenario instances of the proposed framework use the Ryu modular SDN controller. Each module of the Ryu controller listens for specific events and catalogues the same. Ryu has an event processor, which dequeues the received event queues and calls the appropriate event handler. The proposed framework has a module that tracks the LACP state based on LACP data units exchanged. A module that maintains links utilization and ports status. Another module processes application requests for validity based on network snapshot information and administration policies and evaluates whether the request complies with its criteria. A final module implements the network flow behavior in the data plane.

Simulation Setup

Two network topologies were created in Miniedit for the experiment, as shown in Figures *3* and 4. Figure *3* had two link aggregation groups and Figure *4* had four link aggregation groups. Both topologies have the same LACP functions configuration. Thus, the proposed algorithm was evaluated with both topologies. This was to access the extent to which the number of LA groups in an LACP influences bandwidth awareness in a heavy traffic network.

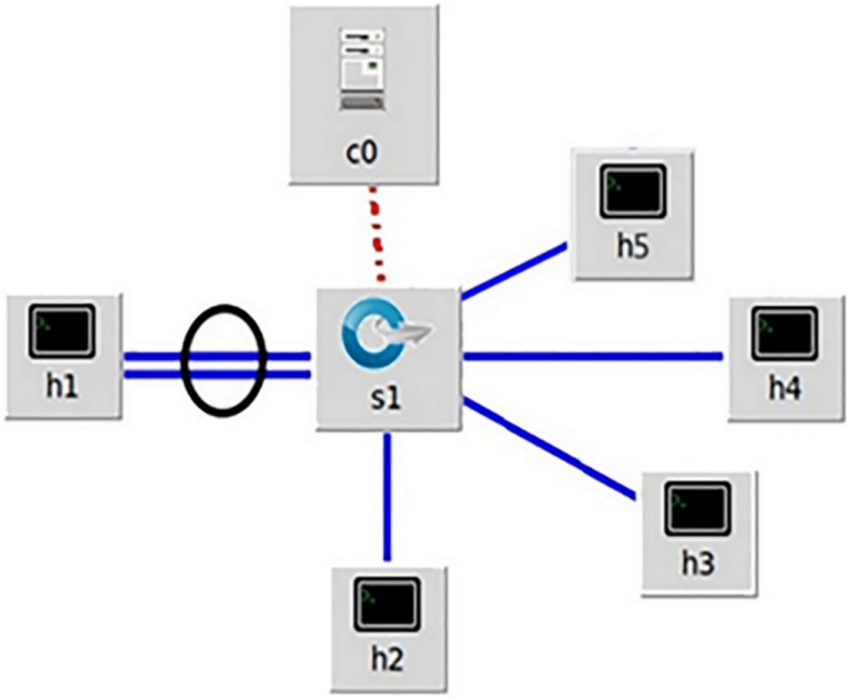

FIGURE 3: Two-link LAG topology

Source: Prepared by the authors based on Romanov et al. [32]

LAG, Link Aggregation Group

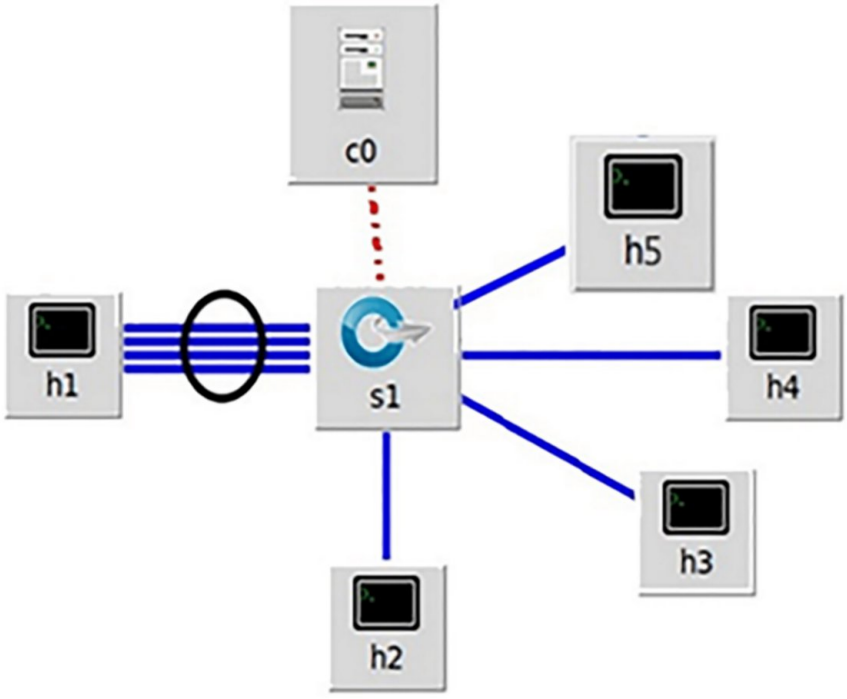

FIGURE 4: Four-link LAG topology

Source: Prepared by the authors based on Romanov et al. [32]

LAG, Link Aggregation Group

Simulation and Evaluation of the Proposed Algorithm

Firstly, the data plane, control plane elements, and network topology were programmatically created in python3 and were saved in two .py files. The .py file for the network topology was run in the ubuntu console in an activated mode using the ubuntu script “sudo python3 topo.py” (where topo is the name of the topology file created in python3). Further configurations were done on the various devices, and APIs were installed on the controller. For example, in this experiment, the controller and network behavior were installed on the host named c0 (controller) by running the .py file in the c0 ubuntu console. Also, a simple web server script was run on h1 so that it can serve as the LMS server in this experiment. The last step in the network configuration was the creation of the virtual link using Linux bonding and then setting the virtual link (bond0) up as the master link for data transmission and the physical links as slaves to be used for failover.

Test traffic was generated to measure bandwidth, latency, and fault tolerance. The testing process was repeated using iPerf3 and ping (ICMP) until the system continuously produced the same results three consecutive times. The system was evaluated using the values obtained for bandwidth, latency, packets loss, and fault tolerance during simulation for both TCP and UDP packets.

The bandwidth and latency were observed as TCP and UDP iPerf3 packets were transmitted between the LMS server (h1) and the connected hosts (h2, h3, h4, and h5). The server-side iPerf3 was run on h1 (set up as the LMS server), and the client-side iPerf3 was run from h2, h3, h4, and h5, respectively. Individual and simultaneous client TCP and UDP iPerf3 packets were transmitted from h2, h3, h4, and h5 to measure the bandwidth of the connections to the server (h1).

Each host (h2, h3, h4, and h5) took turns transmitting 10,000 MB TCP and UDP iPerf3 packets to the server (h1) for 15 seconds, and their respective bandwidths were recorded. Afterward, all the connected hosts (h2, h3, h4, and h5) simultaneously transmitted 10,000 MB TCP and UDP iPerf3 packets to the server (h1), and the bandwidth, latency, and packet losses were recorded.

After collecting data on how effectively the LACP application influenced bandwidth reservation and QoS

of the LMS, the system was reconfigured by deleting the master status of the virtual link (bond0) to check how the system responds to faults. In this case, traffic was channeled through the available physical links to the LMS server (h1), and the bandwidth was analyzed again.

Results

Bandwidth

The system was evaluated by transmitting 10,000 MB of iPerf3 UDP datagrams and TCP streams at a packet rate of 10,000 packets/s for 15 seconds. The same amount of TCP and UDP packets were transmitted at the same packet rates from the connected hosts (h2, h3, h4, and h5) to the server (h1).

The results of the simulation show that the system dynamically allocates an average bandwidth of 11,900.4 Mbps for each stream on all the hosts involved in the simultaneous UDP datagram stream transmission. The two-LAG LACP experiment showed much more stable bandwidth reservation, as shown in Figure *5*. However, the four-LAG LACP experiment showed a wider bandwidth reservation across the hosts and by datagram packet rate, as shown in Figure *6*.

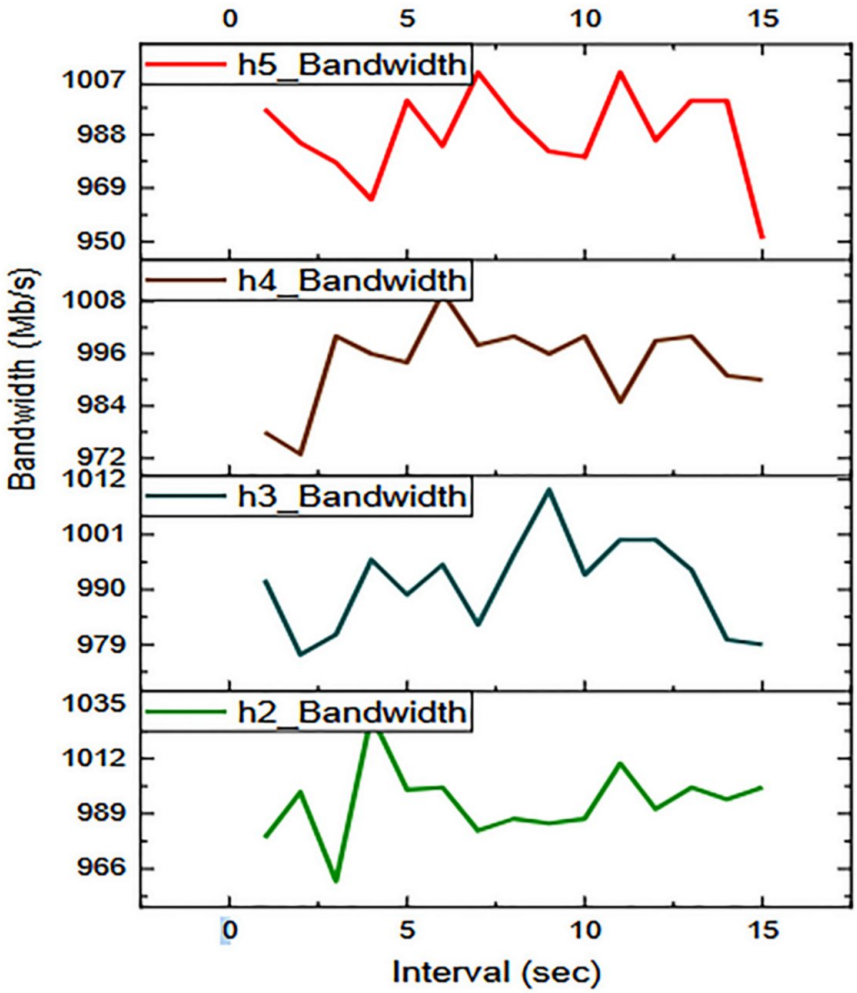

FIGURE 5: Bandwidth of iPerf3 UDP stream datagrams in a two-LAG setup

LAG, Link Aggregation Group; UDP, User Datagram Protocol

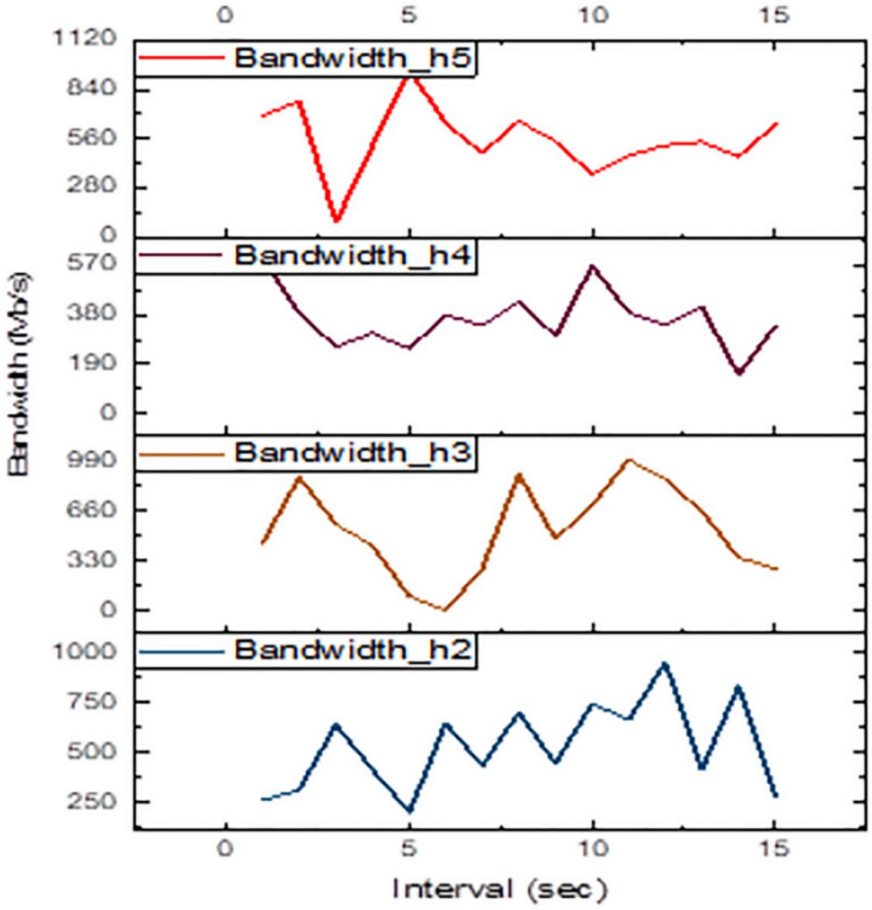

FIGURE 6: Bandwidth of iPerf3 UDP datagrams in a four-LAG setup

LAG, Link Aggregation Group; UDP, User Datagram Protocol

The system behaves differently when TCP ICMP packets were transmitted in both two-LAG and four-LAG experiments. The setup with two LAGs was consistent in allocating nearly 100% of the reserved bandwidth. Only a 0.2% drop in bandwidth allocation was recorded for some TCP flood streams, as shown in Figure *7*.

The four LAG system allocated bandwidth dynamically to enhance a stable flow of TCP flood streams, as shown in Figure *8*. However, it was observed that the system uses 80% of CPU resources to arrive at this stability. Additionally, a marginal increase in interpacket gaps (IPGs) and packet retransmission was occasioned by the system's inability to use all of the allocated aggregated links bandwidth.

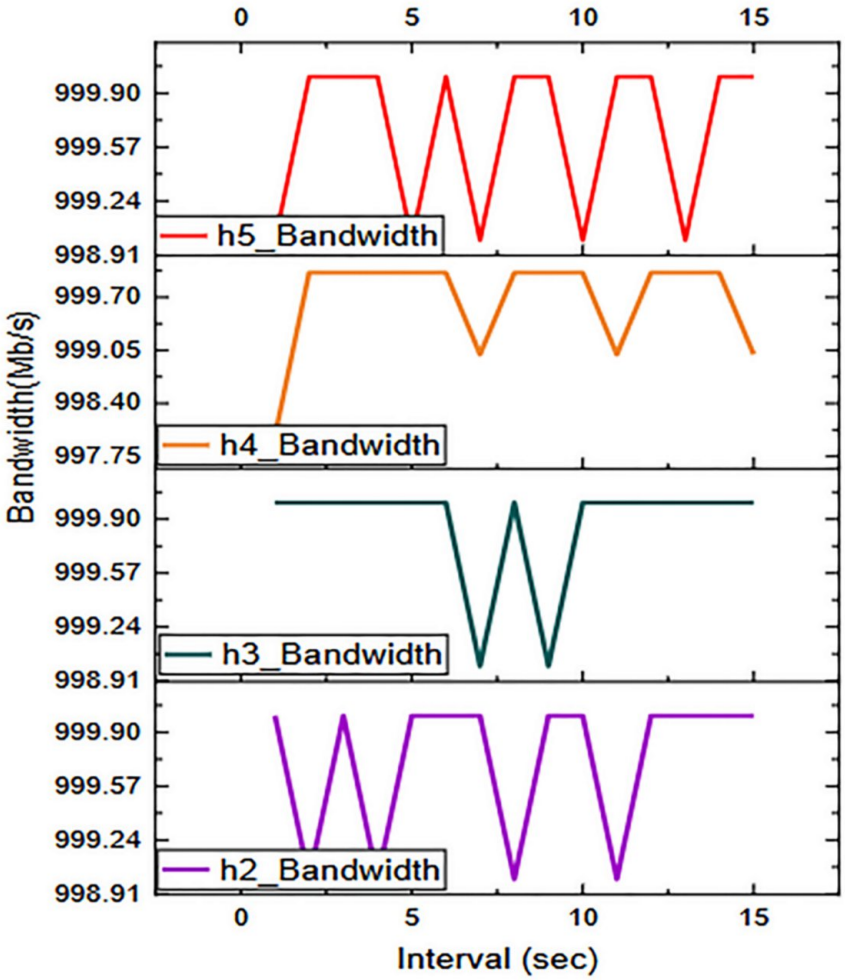

FIGURE 7: Bandwidth of iPerf3 TCP packets in a two-LAG LACP setup

LACP, Link Aggregation Control Protocol; LAG, Link Aggregation Group; TCP, Transmission Control Protocol

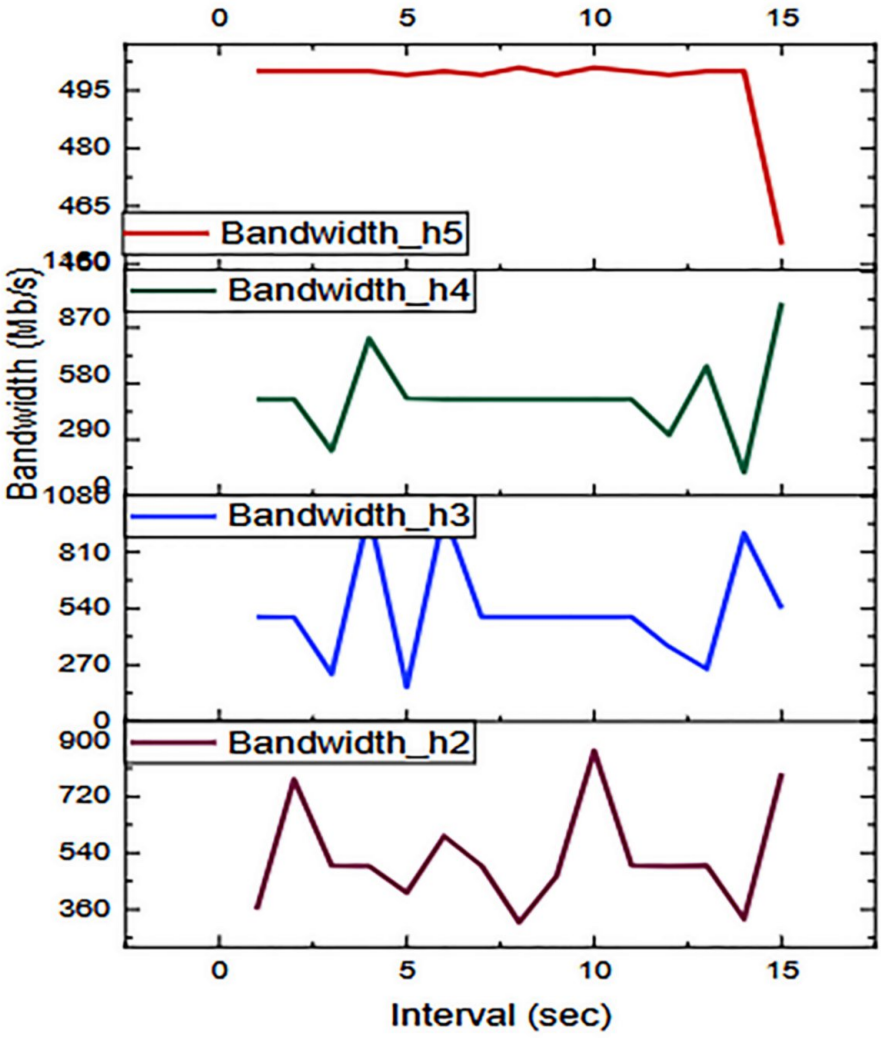

FIGURE 8: Bandwidth of iPerf3 TCP ICMP packets in a four-LAG LACP setup

ICMP, Internet Control Message Protocol; LACP, Link Aggregation Control Protocol; LAG, Link Aggregation Group; TCP, Transmission Control Protocol

Latency

A total of 1,000 Mb of ping flood packets were simultaneously transmitted from the connected hosts (h2, h3, h4, and h5) to the server (h1) over a duration of 100 seconds. The minimum latency, average latency, maximum latency, mean deviation, interframe spacing, and exponentially weighted moving average values were recorded, as shown in Table *1*.

	Min Latency (ms)	Avg Latency (ms)	Max Latency (ms)	Mdev (ms)	IPG (ms)	EWMA (ms)
h2	0.007	0.093	5.398	0.612	0.11	0.008
h3	0.009	0.102	4.89	0.557	0.124	0.009
h4	0.006	0.499	31.849	3.467	0.374	0.008
h5	0.007	0.113	6.024	0.695	0.135	0.013

TABLE 1: A summary of latency report for 100 s of ping ICMP flood in a two-LAG LACP setup

EWMA, Exponentially Weighted Moving Average; ICMP, Internet Control Message Protocol; LACP, Link Aggregation Control Protocol; LAG, Link Aggregation Group; IPG, Interpacket Gap

The results of the ping flood testing, as shown in Table *1*, indicate that it took an average of 0.807 ms to transmit a total of 400 ICMP packets in a simultaneous TCP flood of 4,000 Mb of data transmitted, an indication of fast end-to-end latency. The average latency for h4 was the highest of all the hosts because there was a relatively high IPG observed. This resulted in a delay in the delivery of packets in an end-to-end communication for h4. However, this IPG was an isolated case and could not result in the transmission of duplicate or out-of-order packets.

Table *2* shows the average latencies for simultaneous transmission of 1,000 M of TCP flood requests and acknowledgments for each connected host (h2, h3, h4, and h5) and the server (h1) for 100 seconds.

	Min Latency (ms)	Avg Latency (ms)	Max Latency (ms)	Mdev (ms)	IPG (ms)	EWMA (ms)
h2	0.007	0.551	13.162	2.292	0.237	0.01
h3	0.006	0.084	3.224	0.433	0.103	0.008
h4	0.01	0.134	6.774	0.814	0.172	0.024
h5	0.009	0.143	8.201	0.89	0.178	0.015

TABLE 2: A summary of ping flood results for each host for 100 s in a four-LAG LACP setup

EWMA, Exponentially Weighted Moving Average; LACP, Link Aggregation Control Protocol; LAG, Link Aggregation Group; IPG, Interpacket Gap

Fault tolerance

The Linux command (ip link set h1-eth0 nomaster) was run on the server (h1) to reconfigure the network setup for testing fault tolerance. This command deletes the logical link (bond0), leaving only the physical link that connects h1-eth1 available for data transmission since the physical link that connects h1-eth0 was deleted after setting up the logical link (bond0). In the two-LAG LACP experiment, when the h1-eth0 link was deleted, the setup had h1-eth1 as the link available for data communication. The system was tested by transmitting 1,000 M UDP flood from the hosts (h2, h3, h4, and h5) to the server (h1). Table *3* shows a summary report of the systems' response for bandwidth, jitter, and packet loss rate.

Hosts	Bandwidth (Mb/s)	Jitter (ms)	Packets Loss Rate (%)
h2	549	0.001	9.7
h3	555	8.027	9.5
h4	603	0.005	17
h5	647	0.001	6

TABLE 3: Fault tolerance in a two-LAG LACP

LACP, Link Aggregation Control Protocol; LAG, Link Aggregation Group

The results show a decline in the bandwidth of the available link as compared to the bandwidth of the logical link. However, fewer packets were lost in transmission due to the less time taken by the system to recover from the sudden disconnection of the logical link as shown in the low jitter values recorded.

Similarly, the command (ip link set h1-eth0 nomaster) run on the server deletes one link (h1-eth0) for the four-LAG experiment. This means that three more links (h1-eth1, h1-eth2, and h1-eth3) were made available for data transmission. The result did not show any significant impact on the bandwidth, jitter, or packets loss rates, as seen in Table *4*. Therefore, the command (ip link set h1-eth1 nomaster) was run again to delete h1-eth1, leaving h1-eth2 and h1-eth3 links up for data transmission. The system was tested again by transmitting 1,000 M UDP flood from the hosts (h2, h3, h4, and h5) to the server (h1). Table *4* displays the summary report of the systems' response for bandwidth, jitter, and packet loss rate.

Hosts	Bandwidth (Mb/s)	Jitter (ms)	Packets Loss Rate (%)
h2	61.8	0.184	82
h3	372	0.717	7.4
h4	242	0.12	33
h5	500	0.001	0.074

TABLE 4: Fault tolerance in a four-LAG LACP

LACP, Link Aggregation Control Protocol; LAG, Link Aggregation Group;

The results in Table *4* show a faster bandwidth depletion as more links were deleted and UDP flood datagrams keep transmitting at the same packet rate. This was due to the longer time and the high CPU resources that were used in recovering from the link loss and packet retransmission.

Comparing the results from Tables *3* and 4 indicates that the Two-LAG LACP setup was more efficient in fault tolerance, system resources consumption, and bandwidth reservation than the four-LAG LACP setup.

Discussion

Bandwidth

From Figure *5*, bandwidth was allocated to each datagram stream proportionally to the amount of data transmitted. The average bandwidth observed over 15 seconds of data transmission was 11,900.4 Mbps. This represents approximately a 60.5% improvement in bandwidth reservation for the self-hosted system compared to the results reported by Irawati et al. [12], which recorded an average bandwidth of 7,413.33 Mbps over 600 seconds of data transmission.

Figure *6* shows a maximum reservation of network bandwidth for all streams of ICMP flood transmitted simultaneously from the connected hosts to the server. The lowest bandwidth observed for a packet stream is 998 Mbps, and the largest available bandwidth is 1,000 Mbps, thus providing a reliable bandwidth assurance, making it possible to transmit about 98% of ICMP streams.

Latency

From Table *1*, the results of the ping flood testing show that it took an average of 0.807 ms to transmit a total of 400 ICMP packets in a simultaneous TCP flood of 4,000 Mb of data transmitted, an indication of a fast end-to-end latency. The average latency for h4 was the highest of all the hosts because there was a relatively high IPG observed. This results in a delay in the delivery of packets in an end-to-end communication for h4. However, this IPG was an isolated case and could not result in the transmission of duplicate or out-of-order packets.

The results in Table *2* show that it took an average of 0.032 ms to transmit a total of 400 ICMP packets in a simultaneous TCP flood of 4,000 Mb of data transmitted. This indicates that when the number of aggregated links was increased, it takes a much shorter time for end-to-end communication to be established when transmitting ICMP packets. However, duplicated acknowledgements were received for some ping requests, which accounts for the inching up observed in the IPG for all the transmitting hosts. This was because acknowledgment for some packets was duplicated.

Fault tolerance

The results in Table *3* show a decline in the bandwidth of the available link as compared to the bandwidth of the logical link. However, fewer packets were lost in transmission due to the less time taken by the system to recover from the sudden disconnection of the logical link as shown in the low jitter values recorded.

The results in Table *4* show faster bandwidth depletion as more links were deleted and UDP flood datagrams keep transmitting at the same packet rate. This was due to the longer time and the high CPU resources, which are used in recovering from the link loss and packet retransmission.

Limitations

The proposed system is limited by several factors, including high CPU saturation observed in the four-LAG setup, issues with fixed thresholds, and challenges with the Ryu controller. The system utilized about 75% of CPU resources to maintain stable TCP flood stream transmission and about 80% of CPU resources to maintain stable UDP flood datagrams. This led to increased IPGs, packet retransmission rates, and packet losses during UDP datagram transmission. In addition, the use of a fixed 80% bandwidth threshold to trigger link aggregation did not adapt well to the highly variable traffic patterns used during the simulation. This resulted in poor implementation of port monitoring and aggregation functions, affecting the efficient use of links. Furthermore, the Ryu controller exhibited inefficiencies in clearing the LAG logs, which impacted bandwidth reassignment; hence, increased packet retransmission rates were noted as the number of aggregated links increased.

Conclusions

This paper presented a framework for dynamic bandwidth allocation for self-hosted educational systems. The LACP framework is configured to dynamically allocate bandwidth and was tested in two-LAG and four-LAG environments. The proposed dynamic bandwidth-aware LACP algorithm for self-hosted LMS data centres was effective in dynamic allocation of bandwidth based on traffic demands. The results showed that the LACP system was cheaper but efficient in adjusting link bandwidth and dynamically allocating same in response to the traffic density of the educational system at any given time. In the two-LAG LACP environment, for instance, as bandwidth allocation increased, fault tolerance and the QoS of the system also increased accordingly. On the other hand, the four-LAG LACP environment showed a slight decline in fault tolerance and QoS of the system. Interpacket delays and packets retransmission rate was increased, resulting in high CPU utilization for path negotiation and renegotiation.

To resolve these challenges with the increasing links bundling, further studies would be conducted to include to include energy efficient metrics, including controller capabilities in future dynamic bandwidth-aware LACP implementation. With this framework, network administrators can develop more efficient APIs to inform the SDN controller of their specific requirements for data flow contexts over any chosen number of aggregated links.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Bright K. Kunu, Joshua C. Dagadu

Acquisition, analysis, or interpretation of data: Bright K. Kunu, Stephen K. Pometsey, Oliver K. Boansi

Drafting of the manuscript: Bright K. Kunu

Critical review of the manuscript for important intellectual content: Bright K. Kunu, Stephen K. Pometsey, Oliver K. Boansi, Joshua C. Dagadu

Supervision: Joshua C. Dagadu

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** The article is previously published as a preprint on https://www.researchsquare.com/article/rs-3453382/v1.

References

1. Heller B, Scott C, McKeown N: Leveraging SDN layering to systematically troubleshoot networks. HotSDN '13: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. 2013, 37-42. 10.1145/2491185.2491197
2. Zhao H, Niu W, Qin Y, Ci S, Tang H, Lin T: Traffic load-based dynamic bandwidth allocation for balancing the packet loss in DiffServ network. 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, Shanghai, China. 2012, 99-104. 10.1109/ICIS.2012.113
3. Wang C, Shanbhag S, Wolf T: Virtual network mapping with traffic matrices. 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada. 2012, 2717-2722. 10.1109/ICC.2012.6364255
4. Suchara M, Xu D, Doverspike R, Johnson D, Rexford J: Network architecture for joint failure recovery and traffic engineering. ACM SIGMETRICS Performance Evaluation Review. 2011, 39:97-108. 10.1145/2007116.2007128
5. Alsaeedi M, Mohamad MM, Al-Roubaiey AA: Toward adaptive and scalable OpenFlow-SDN flow control: A survey. IEEE Access. 2019, 7:107346-107379. 10.1109/access.2019.2932422
6. Feamster N, Borkenhagen J, Rexford J: Guidelines for interdomain traffic engineering. ACM SIGCOMM Computer Communication Review. 2003, 33:19-30. 10.1145/963985.963988
7. Zhang N, Yang P, Zhang S, Chen D, Zhuang W, Liang B, Shen XS: Software defined networking enabled wireless network virtualization: Challenges and solutions. IEEE Network. 2017, 31:42-49. 10.1109/mnet.2017.1600248
8. Han Y, Li J, Hoang D, Yoo JH, Hong JWK: An intent-based network virtualization platform for SDN. 2016 12th International Conference on Network and Service Management (CNSM), Montreal, QC, Canada. 2016-10, 353-358. 10.1109/CNSM.2016.7818446
9. Narmanlioglu O, Zeydan E: Software-defined networking based network virtualization for mobile operators. Computers & Electrical Engineering. 2017, 57:134-146. 10.1016/j.compeleceng.2016.09.011
10. Barona López LI, Valdivieso Caraguay ÁL, García Villalba LJ, López D: Trends on virtualisation with software defined networking and network function virtualisation. IET Networks. 2015, 4:255-263. 10.1049/iet-net.2014.0117
11. Petrosino A, Sciddurlo G, Grieco G, Shah AA, Piro G, Grieco LA, Boggia G: Dynamic management of forwarding rules in a T-SDN architecture with energy and bandwidth constraints. Ad-Hoc, Mobile, and Wireless Networks. Grieco LA, Boggia G, Piro G, Jararweh Y, Campolo C (ed): Springer, Cham; 2020. 12338:3-15. 10.1007/978-3-030-61746-2_1
12. Irawati ID, Hariyani YS, Hadiyoso S: Link aggregation control protocol on software defined network. International Journal of Electrical and Computer Engineering. 2017, 7:2706-2712. 10.11591/ijece.v7i5.pp2706-2712
13. Nurhadi AI, Petrus GBK, Firdaus M, Muhammad R: A review of link aggregation control protocol (LACP) as a link redundancy in SDN based network using Ryu-controller. ArXiv. 2020, 10.48550/arXiv.2005.14652
14. Shurygin V, Saenko N, Zekiy A, Klochko E, Kulapov M: Learning management systems in academic and corporate distance education. International Journal of Emerging Technologies in Learning (iJET). 2021, 16:121-139. 10.3991/ijet.v16i11.20701
15. Almaiah MA, Al Mulhem A: A conceptual framework for determining the success factors of E-learning system implementation using Delphi technique. Journal of Theoretical and Applied Information Technology. 2018, 96:5962-5976.
16. Qu Y, Chen IH, Meng S: Multimedia teaching resource allocation method for distance online education based on packet cluster mapping. International Journal of Information and Communication Technology. 2023, 23:314-326. 10.1504/ijict.2023.134841
17. Abdulmohson A, Kadhim MF, Anssari OMH, Al-Jobouri AA: Cost analysis of on-premise versus cloud-based implementation of moodle in Kufa University during the pandemic. Indonesian Journal of Electrical Engineering and Computer Science. 2022, 25:1787-1794. 10.11591/ijeecs.v25.i3.pp1787-1794
18. Ekuase-Anwansedo A, Smith A: Effect of cloud based learning management system on the learning management system implementation process. SIGUCCS '19: Proceedings of the 2019 ACM SIGUCCS Annual Conference. 2019, 176-179. 10.1145/3347709.3347835
19. Sokol VY, Sapronov PY, Bilova MO: Using cloud platforms to build distributed learning management systems. Bulletin of National Technical University "KhPI". Series: System Analysis, Control and Information Technologies. 2020, 2:33-38. 10.20998/2079-0023.2020.02.06

20. Angeli L, Huang Y, Marchese M: Conceptualising resources-aware higher education digital infrastructure through self-hosting: a multidisciplinary view. Eighth Workshop on Computing within Limits 2022, LIMITS. 2022, 1-12. 10.21428/bf6fb269.8b989f2c
21. Latif Z, Sharif K, Li F, Karim MM, Biswas S, Wang Y: A comprehensive survey of interface protocols for software defined networks. Journal of Network and Computer Applications. 2020, 156:102563. 10.1016/j.jnca.2020.102563
22. Gharbaoui M, Martini B, Castoldi P: Programmable and automated deployment of tenant-managed SDN network slices. NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary. 2020, 1-6. 10.1109/NOMS47738.2020.9110302
23. Benzekki K, El Fergougui A, Elalaoui AE: Software-defined networking (SDN): a survey. Security and Communication Networks. 2016, 9:5803-5833. 10.1002/sec.1737
24. Sezer S, Scott-Hayward S, Chouhan PK, Fraser B, Lake D, Finnegan J: Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Communications Magazine. 2013, 51:36-43. 10.1109/mcom.2013.6553676
25. Yen TC, Su CS: An SDN-based cloud computing architecture and its mathematical model. 2014 International Conference on Information Science, Electronics and Electrical Engineering, Sapporo, Japan. 2014, 1728-1731. 10.1109/InfoSEEE.2014.6946218
26. Mishra S, Alshehri MAR: Software defined networking: research issues, challenges and opportunities. Indian Journal of Science and Technology. 2017, 10:1-9. 10.17485/ijst/2017/v10i29/112447
27. IEEE 802.1AX-2020. IEEE Standard for Local and Metropolitan Area Networks--Link Aggregation. (2020). https://standards.ieee.org/ieee/802.1AX/6768/.
28. Baskoro F, Hidayat R, Wibowo SB: LACP experiment using multiple flow table in Ryu SDN controller. 2019 2nd International Conference on Applied Information Technology and Innovation (ICAITI), Denpasar, Indonesia. 2019, 51-55. 10.1109/ICAITI48442.2019.8982149
29. Khan R, Ali S: Conceptual framework of redundant link aggregation. Computer Science & Engineering: An International Journal. 2013, 3:23-31. 10.5121/cseij.2013.3202
30. Ferguson AD, Guha A, Liang C, Fonseca R, Krishnamurthi S: Participatory networking: an API for application control of SDNs. ACM SIGCOMM Computer Communication Review. 2013, 43:327-338. 10.1145/2534169.2486003
31. Lee J, Turner Y, Lee M, Popa L, Banerjee S, Kang JM, Sharma P: Application-driven bandwidth guarantees in datacenters. SIGCOMM '14: Proceedings of the 2014 ACM conference on SIGCOMM. 2014, 467-478. 10.1145/2619239.2626326
32. Romanov O, Nesterenko M, Marinov A, Skolets S, Burlaka H: SDN network modeling using the GUI MiniEdit. 2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine. 2022, 01-06. 10.1109/TCSET55632.2022.9766897