The Journal of Supercomputing
https://doi.org/10.1007/s11227-022-04929-y

Efficient bandwidth allocation in SDN-based peer-to-peer data streaming using machine learning algorithm

Hamza Aldabbas[1]

Accepted: 28 August 2022

Abstract

Software-defined networking (SDN) is a contemporary structural design paradigm that aspires to correct bandwidth-efficient usage and user application transparent interoperability. It claims to be self-motivated, convenient, affordable, and programmable. The network control may become directly programmable thanks to SDN architecture's decoupling of the network control and management plane from the data plane. Due to the centralized architecture of the SDN network, message propagation to various network devices in SDN is delayed. The SDN controller must thus establish new rules for each new communication. Peer-to-peer protocols are deployed in the system for the fast propagation of contents over the layered architecture. However, these protocols produce unwanted packets because these protocols mostly use a reactive routing protocol over an SDN architecture. This causes the delay for different applications due to inefficient bandwidth management. By offering resistance to connection failures and accommodating constantly changing bandwidth needs, it is difficult to control the needed bandwidth. This paper proposes a new technique using the Random Forest algorithm for efficient bandwidth management for peer-to-peer applications. In this technique, bandwidth usage of different applications is computed and predicted. So, according to the need of applications, network traffic is adjusted accordingly. An ultra-peer node is responsible for communication with other nodes in this methodology, eliminating unwanted traffic across the network. A Linux-based OpenvSwitch, POX controller and Mininet-based SDN-based network system are used for the tests. The results of the experiment demonstrate that the proposed framework may significantly improve QoS while outperforming the current bandwidth allocation algorithms in terms of success rate, throughput, response time, etc.

Keywords Peer-to-peer protocol (P2P) · Software-defined network (SDN) · Gnutella (GN) · BitTorrent (BT) · OpenFlow (OF)

Hamza Aldabbas
aldabbas@bau.edu.jo

Extended author information available on the last page of the article

1 Introduction

Software-defined networking (SDN) performs administration adaptability and control by providing programmability modules for system manipulation [1, 2]. These modules are critical to empowering the framework to react rapidly and have the ability to apply changes to the underlying topology when it is needed. This programmability segment is the clear separation of the data plane, connected with sending packets, from the control plane, related to coordinating decisions [3]. SDN is a cutting-edge paradigm that claims to be naturally awakened, helpful, and adaptive. It aims to be suitable for the dynamic, high-transfer-speed nature of current applications. By separating the transmission and control functions of the system, SDN architectures enable the system control to become purely programmable [4]. SDN is an innovative technology to overcome the loophole of the traditional network along with its continuously architectural development for higher adaptability. One problem is the slow propagation of messages during the communication among SDN and overlays architecture for searching or sharing the file between the networks [5].

The primary motivation behind this programmability system is the reasonable division of the data plane from the control plane [6]. Thus, SDN comprises three planes providing a promising environment for the end user at low cost, open for the customized network application, network orchestration, and global network view at the central point of the network. Management plane's applications interact with the control plane via northbound API and control plane with forwarding devices via southbound API (OpenFlow). Figure 1 presents the SDN planes and their interactions broadly [7].

The engineering of SDN is the layered design, and the correspondence among the associated layers is the basic operation that facilitates the administrations. The spread of messages to various system gadgets is easing the system because of the unique nature of SDN [8]. In this regard, well-known peer-to-peer protocols like BitTorrent are used for fast propagation of content. SDN uses the reactive routing protocol, and BitTorrent is deployed as a peer-to-peer protocol for the fast propagation of contents over the layered architecture. Unfortunately, this mechanism produces unwanted packets due to the underlying SDN architecture [9]. This work sorts the cooperation issues into three classes: fully internal, fully external, and half internal, relying upon the area of the members within the neighborhood. When we frequently search for the best path then saturate each channel, so the BitTorrent choking strategies are the sense of the change of underlying topology. In reactive routing, the changes are occurred in top layer traffic, representing the overall point of reactive routing [10].

This paper proposes to contemplate the impact of SDNs' reactive routing considerations on various development circumstances for the peer-to-peer network. Using the Gnutella protocol, a new mechanism was introduced in which peers are grouped; then, the head peer is chosen. In order to manage the network bandwidth and traffic efficiently, a machine learning algorithm random forest is deployed to predict the network traffic load on different links. The head peer communicates

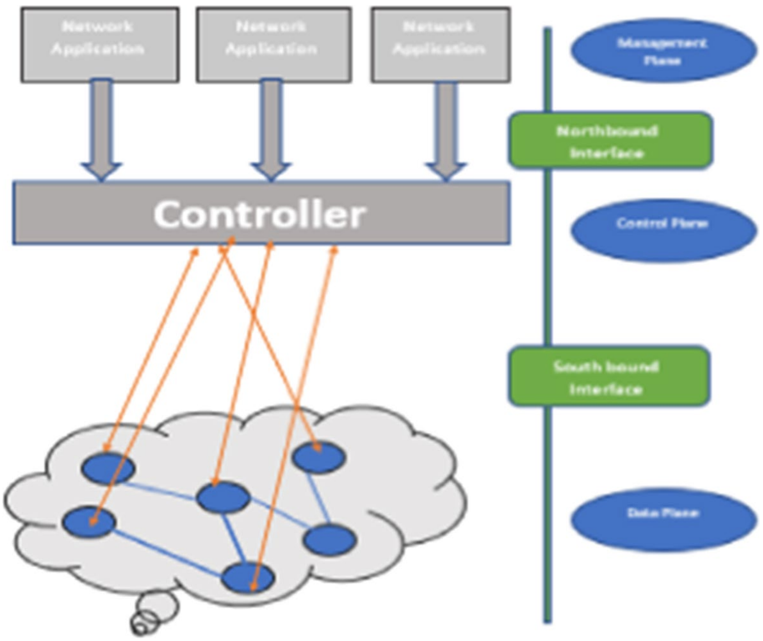

Fig. 1 SDN Architecture

with other peers and helps in managing the bandwidth allocation for different applications using SDN. In this way, the entire network performed efficiently. The efficiency of the proposed mechanism was assessed using simulation results. The main contribution of this paper is as follows:

- The study of the deployment of peer-to-peer network applications over an SDN-based network and analyzed the link bandwidth and latency issues.
- Proposing a machine learning-based solution to solve the issue of link bandwidth related to the dynamic bandwidth allocation according to the need of the applications.
- In this proposed mechanism, the introduction of ultra-peer nodes responsible for communication with other nodes, thus eliminating unwanted traffic across the network.
- A machine learning algorithm random forest is deployed to predict the network traffic load on different links. In this way, the traffic is adjusted according to the need of the applications and links.
- The proposed mechanism achieves a high response rate, low overhead, high success rate, etc., over the existing mechanism.

The rest of the paper is organized as follows: Section 2 describes the related work and problem formulation is elaborated in Sect. 3. The proposed solution is discussed in Sect. 4 with details of the Random Forest algorithm to manage the link bandwidth. Section 5 includes the implementation details and performance evaluation of the system and Sect. 6 concludes the paper.

2 Related work

SDN is yet a rising innovation, and the pace of SDN innovation progress meant is advancing quickly [11]. One of SDN's key points of interest is that SDN enables application-level control/programming of the framework. In [12], a new network controlling mechanism for a large-scale enterprise computer network is introduced. This mechanism can take the place of the current network configuration, control, and management mechanism. Network management, however, is a big issue due to the increase in wireless and wired users worldwide. About 80% of company budgets will be spent on maintenance, control, and configuration of the networks. One solution is by adding an extra layer of protocols and applications. This solution is based on the 4D [13] project to adopt central control management. It also resembles SANE [14], a clean slate solution for enterprise security.

Ethane [12] is developed given three principles: first, a network should be built using policies and based on entities. Second, these policies should determine the path of the packet flow. Third, there must be a strong binding between the packet and its origin. Ethane contains the main controller with a global network policy describing the path for data packets. This controller has all information about network components and finds routes for permitted flows. Ethane consists of several Ethane switches which are more straightforward and dumber. These switches include a flow table and a secure channel to the controller. When a packet arrives, and the switch does not have a path for it, it merely forwards it to the controller with port number. The controller keeps the information for entities, hosts, and devices in the network and keeps mapping information for machines, IP addresses, IP addresses, and MAC addresses. As the controller has a centralization mechanism, it is easy to keep the namespace consistent as node leave joins or moves to some other space.

Another approach, the Application Layer Trace Optimization (ALTO) [15], gives a benchmarks-based plan in which SDN can deploy different applications, including Peer-to-peer network applications. For example, by using the proposed ALTO SDN approach, a BitTorrent client can observe the activities of a tracker (middle person) who uses ALTO SDN to keep up the information UpToDate [16]. Accordingly, SDN may not over favorable critical position appeared differently about the existing application-level partner assurance frameworks [17].

SDNs' advances show intuitive ways to overhaul the framework topology; one possibility of using SDNs is through Reactive Routing [18]. On Reactive Routing, the sensible topology is unendingly creating in light of follow estimations, such as workload or jitter, accumulated by the switches. Despite the way that layer exemplification should roll out these improvements in the lower layers direct to the Application Layer, the fact of the matter is far from that for this circumstance [19]. The most famous cases are Pareto proficiency and BitTorrent choking estimations [20]. The measure of P2P movement and the number of inhabitants in P2P clients on the Internet continues expanding. A considerable test of studies has been performed on the estimations, displaying, and calculations of various P2P frameworks. The creators additionally found that the entry, prematurely ends, and flight procedures of downloaders try not to take after a Poisson dispersion in the 8-month follow they gathered, which was expected in the past demonstrating study [21]. The administration limit of BitTorrent-like frameworks and found that multipart downloading causes P2P frameworks to enhance execution amid a streak swarm period.

BitTorrent convention was mostly planned to be a substitute for old concentrated HTTP downloads and not a full p2p convention, as shown in Fig. 2. As it developed, it progressed toward becoming distributed in nature where clients associated with each other straightforwardly to transfer and download segments of a vast document (called as a piece) from different companions who have likewise downloaded either the record or parts of it [22]. Then reassembled into the full record. This instrument gave speedier download though, in typical p2p frameworks, the companion downloads from a solitary associate alone and the download speed are restricted. The traditional brought together server-based types of downloads were defaced with server bottlenecks and added to arrange clog. While BitTorrent was intended to give a quicker download and clog free component. The property of concurrent transferring while simultaneously downloading guarantees that there is no single source to download from (along these lines expelling the bottleneck issue) and guarantees that system movement because of record download gets disseminated over the system [23].

Overlay networks are mainly used for searching and distributing data by using routing and forwarding over an ordinary network [24]. There are large numbers of network designs for overlays network, and in these designs, Distributed Hash Table (DHT) has some special properties and characteristics like scalability and load-balancing capabilities. In the current emerging technologies like Information-Centric Networking (ICN) [25] and Internet of Things (IOT) [26], the network is considered for finding and delivering the different types of contents and services to the user on their demands. DHT provides scalability and other properties based on the logical association of overall network. However, sometimes, this logical organization of

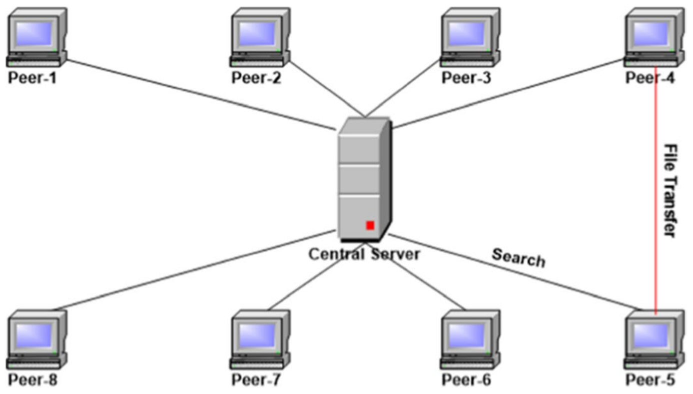

Fig. 2 Centralized peer-to-peer network

nodes in DHT [27] lacks some of the aspects like (1) network administrative domain boundaries, (2) inter-domain routing policies, and (3) physical network proximity. This paper uses a hierarchical pastry H-Pastry, a multi-level DHT scheme that uses the DHT pastry [28] and the canon approach and supports multihoming and peering relationships. This scheme shows that H-Pastry [29] keeps traffic within administrative boundaries and decreases intra-domain hops by up to 27% compared to pastry. It also supports caching and multicast.

Shyan et al. [30] proposed the mechanism of software-defined networking deployment in peer-to-peer networks applications that are sharing data in optical networks. The authors evaluate their architecture as it retains a little bandwidth consumption overhead for passing the P2P wavelengths, but it handles the bandwidth management efficiently in Passive Optical Network (PON) under the collaboration of OpenFlow protocol as shown in Fig. 3. Optical line terminal in PON has multiple optical network unit (ONU), which transmit the data packets using Time and Wavelength multiple access (TWDM). The proposed architecture consists of an SDN controller applications in optical line terminal and agents at ONU, commanded by the controller via a secure channel and OpenFlow protocol to redirect packets with the ONU or intra-ONU transmissions. The P2P controller using the SD-Agent, Mac control, and Flow Table has classified packets based on types and commands the ONUs to modify their wavelength for packet transmission. The ONUs also have the SD-Agents to get an order from optical line terminal controller orders and Flow tables for collaborative transmission of packets and efficient bandwidth utilization.

Although a lot of work is done to mature the SDN, many organizations are still very reluctant to adopt SDN due to several reasons, i.e., critical downtime, untrained staff, more expensive solution, new hazard, and security challenges. A large number of applications are increasing daily, requiring more sophisticated resources and bandwidth provision. An optimal link bandwidth load-balancing approach can be used in networks to utilize optimum energy. The link load method distributes the load by assigning a series of demands to a collection of services. Network upgrades are not only essential in the sense of more dynamic and fine-grained networks (e.g., minimizing the network loads) but also the capability to reroute flows rapidly and reliably is essential for consistency, accessibility, and for overall performance. Network upgrade issues occur because of improvements in the network's policies (security), connection faults, or routine maintenance. A dynamic bandwidth allocation framework, known as DART, is presented

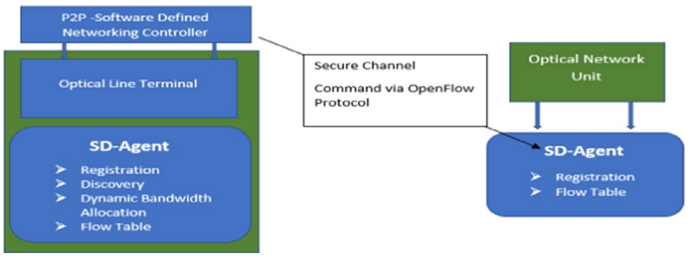

Fig. 3 Optical line terminal interaction with optical network unit and OpenFlow collaboration

in [32] and is capable of dynamically allocating network bandwidth among different services to effectively use the network resources. To reduce congestion on the shared links, the SDN controllers work together in DART to divide the bandwidth among slices. The admission control completes a simple task and provides a priority ceiling. The flow is permitted to continue and a flow rule is sent to the switch each time a new flow seeks bandwidth with a priority higher than the priority limit. Otherwise, the flow is unable to communicate. There are now two different methods for assigning bandwidth to each node of a software-defined network [33, 34]. In contrast to our method, which is flow-oriented, such systems address a node-oriented approach to bandwidth management. A resource provisioning strategy for network flows based on continuous monitoring of a number of thresholds on the range of permitted transmission rates is proposed in the work in [34].

In order to offer effective dynamic resource infrastructure management on Software-Based Networks, this study explores one of the most crucial difficulties in network virtualization [35]. Based on a combination of the Time Division Multiple Access (TDMA) protocol and the Markov-Process, the suggested solution (cTMvSDN) enhances resource management. A customized controller module only starts the mapping when enough resources are available. The future time gaps are predicted using the Markov-Pattern and TDMA slicing model in order to improve reaction time and SDN Quality of service. In TDMA slots, successfully mapped packets will be transmitted. Simulation findings using the NS2 and Mininet simulator indicated an improvement in metrics like latency and expenses.

2.1 Gnutella P2P protocol in SDN

The Gnutella protocol is one of the simple, consistent, and reliable file distribution schemes. Using this mechanism, we can share, distribute and deliver a lot of data like audios, videos, images, multimedia, presentation, documents, etc., to our friends, family members, and other users. This is a distributed system in which no central entity is present to control and manage the network traffic and user communication. All nodes in this network act as peers means every node has the facility to send and receive the files form all other users or peers. These peers work as a server and client at a time and known as servants. Each peer work individually ad independently and collaboratively these peer works as a library of digital contents. This protocol is used for flat routing in peer-to-peer networks [32]. This is a very famous file sharing protocol that spans all over the world. The servants in this protocol are managed by TCP/IP protocol. Gnutella is a traditional protocol used for the scattered request. Gnutella "Request" packages empower you to look for by curious in the matter of whether they are sharing specific data (and have an acceptably brisk framework affiliation) [31].

3 Problem statement

SDN is an innovative technology to overcome the traditional network's loophole, but SDN is still in the development phase. One problem is slow propagation speed during the communication between layered architecture and searching or sharing the file between the network elements.

We show the actual problem using an example scenario shown in Fig. 4. This illustration has three hosts (BT1, BT2, BT3) contributing and these hosts utilize BitTorrent. There are two associations built up among peers. The approach of responsive routing application is that most stream are using the best transfer speed among their peers. Prior to any exchange out of BT1 begins, there is initial synchronization of messages before the connection is built between BT1–BT2 and similar steps are carried out of the BT1–BT3. When selecting the data for exchanging, BT1 consent is to unchoke the BT2, and after that unchokes the BT2. The point of confinement for exchange between these links is 1mbps, and the new connection is established, it can exchange at the speed of 10 Mbps, so the client chose the BT2 as the best connection and then check the BT3 in the following round. As indicated by the arrangement, the controller moves the streams in two ways: b–c and a–c. Therefore, these are utilizing the most conceivable data transfer capacity between peers. Another depraved effect of BitTorrent is that when a few switches exist in the way, a new path needs to be selected every time so that packets can reach the destination within time.

For instance, as an outcome, BitTorrent continues utilizing the least conceivable data transfer capacity between peers. Suppose the reactive routing computes the path before the data transfer. In that case, BitTorrent again picks the wrong one, and BitTorrent never has the ability to get the most extreme possible information exchange. The second issue emerges when numerous switches are in the path from source to destination. The distinction in the paths made by the controller requires the stream tables to get involved for efficient data transmission. This path computation triggers the issue of congestion due to excess data transmission among the nodes.

To lessen the complexity, the interaction can be organized between reactive routing and BitTorrent into three classes: internal, half internal, and fully external.

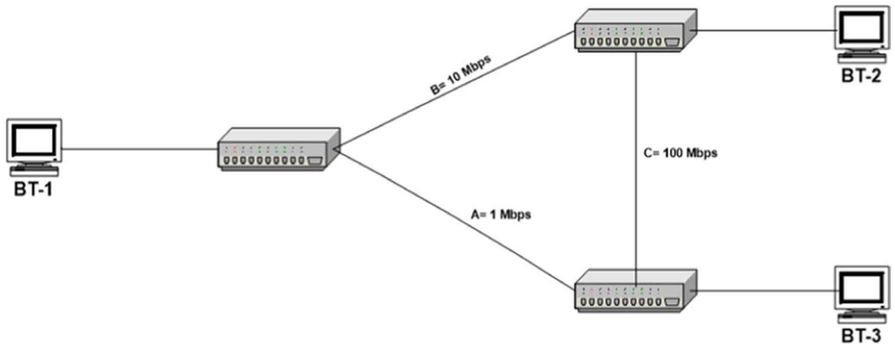

Fig. 4 The network topology in which systems are making an overly peer-to-peer network

While each BitTorrent user is in the identical SDN location, we recall the interaction is fully internal. If one BitTorrent client is within the SDN region and the other is not in the same SDN, it is considered half internal interaction. Finally, while both BitTorrent clients are outside of the SDN region, they are considered fully external. The proposed categories discussed above take one connection between clients; however, the problem is that the BitTorrent might also have three hundred distinctive open connections involved in the same crowd, so the hybrid state of affairs is predicted in exercise.

Today, without going into great depth to optimize bandwidth distribution for each service category, ISPs only limit the overall bandwidth of each residence within subscription bandwidth bounds. For instance, during rush hour traffic, some family members are watching MBA games on YouTube (video streaming), while others are playing online games, using Skype for VoIP chats, browsing Facebook or other websites, or P2P file sharing. It will be challenging to maximize the QoS of delay-sensitive services like video streaming if ISP continues to utilize the conventional approach to manage the bandwidth of a large number of smart homes. Another source of expense is the public disclosure of the network's dynamic features. The link's load and the application's status change over time. The link's load can vary from empty to fully loaded. Furthermore, the applications can begin and stop at any time. Certain measurements are required to convey this information between nearby peers. On the one hand, it would increase the number of messages sent, perhaps resulting in unpredictability. On the other hand, the accuracy of the information exchanged would be difficult to predict.

4 Proposed solution

In SDN, monitoring the network is essential to build a well-working network. All the problems that occurred are needed to be monitored and controlled. For example, peer-to-peer propagation effects when a bandwidth of a defined path varies, and some other existing path keeps a better bandwidth for the same communication. In this regard, communication between two hosts should be interchanged by adopting a better path. In this work, a random forest algorithm is deployed on the SDN centralized unit that computes and predicts a suitable path for the data packets. It instructs the switches to change their paths or routes according to the currently fastest available route for communication.

4.1 Methodology of proposed solution

To fulfill the requirement of the fastest path reaching each time and each packet, we implement a random forest algorithm, considered a good solution for efficient traffic routing. A flexible, simple machine learning method called random forest often produces outstanding results without hyper-parameter modification. It is one of the most popular algorithms because of its simplicity and adaptability (it can be used for classification and regression tasks). To monitor each switch's

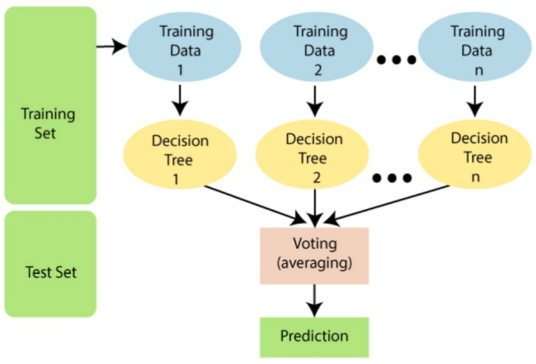

Fig. 5 Working of RF algorithm to predict the possible paths for efficient data transfer

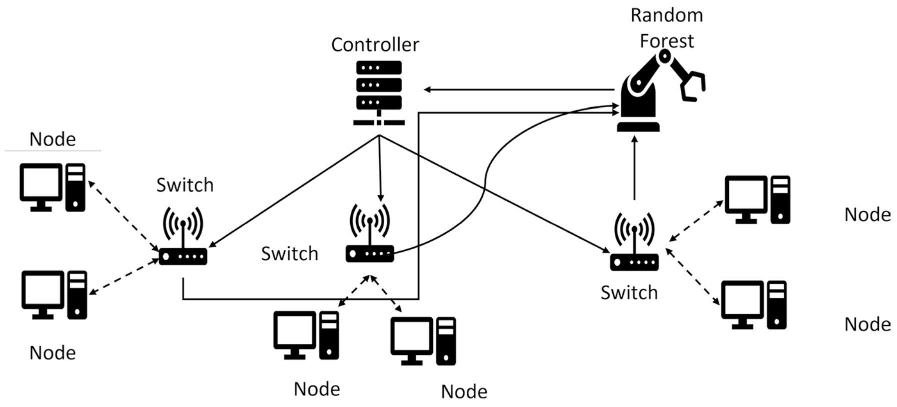

Fig. 6 Diagram of proposed solution

path, queries are acquired periodically from the data plane. Thus, according to updated information gathered from the data plane, the paths are updated periodically. When a decision tree is given a training dataset with features and labels, it generates a set of rules that are used to make predictions, as shown in Fig. 5. Each switch was assigned IDs for each possible path to each switch, which is updated only by exchanging IDs according to the currently fastest available path.

Four switches were introduced into the initial SDN topology in this experiment, and the proposed solution is implemented in the following way, as shown in Fig. 6. First, the nodes are grouped into the form of a cluster—retrieval of the total number of switches as a constant topology. Then, the paths are collected for each switch and assigned IDs to paths switch-wise. The second element is bandwidth collection for each path at a specific time. Finally, this information is plugged into the Random Forest algorithm, a training data set, and after this, test data are used to predict the possible paths for data transfer. In this way, the proposed algorithm worked efficiently.

4.2 Peer-to-peer node management

In this proposal, the controller is in charge of centrally monitoring the P2P streaming network nodes. A content request is made to the controller to find out whether a node has joined or left the P2P streaming network. An OpenFlow switch adjacent to the server receives the flow seen in Fig. 6 and forwards it to the controller. The controller looks through the packet's header when it receives a content request to identify the originating IP address. Following the joining or leaving of the IP address node, the controller modifies the information about the network topology.

4.3 Peer-to-peer network construction and reconstruction

To rebuild a peer-to-peer streaming network, the controller must choose a parent node for a newly connected P2P streaming network node. The controller must also choose a parent node for nodes that were children of a leaving node. The process for determining a parent node is as follows: (1) choose the candidate nodes that are the furthest away from the server and have the most capacity to spare, and (2) out of all the candidates, choose the parent node with the fewest child nodes.

To reconstruct a balanced P2P streaming network, a joining node or child nodes of a leaving node select a parent node using these methods. However, because this approach ignores these nodes' child nodes, the P2P streaming network may be unbalanced. As a result, the controller monitors the topology of the P2P streaming network frequently. If the P2P streaming network is unbalanced, an unbalanced node repeats the above operation to find a parent node. The P2P streaming network is then re-configured to be more balanced.

4.4 Network Topology of Gnutella protocol

Peers preferentially make connections with highly accessible and high out-degree nodes in the overlay in Gnutella, resulting in a vertex connectivity power-law distribution with an index less than 3. In the face of random node failures, such a network is extremely resilient. The overlay, for example, fragments only when more than 60% of the nodes randomly break down. On the other hand, an attack on the overlay's highest degree nodes "shatters" the overlay into a vast number of disconnected components. In addition, firewalls and NATs (Network Access Translation) further complicate network topologies by creating partitioned domains.

Five different message types make up the Gnutella protocol: ping, pong, query, query hit (the response to a query message), and push. In order to find new nodes on the network, a ping message is sent. In response to a ping, a pong message is issued that contains details about a network node, such as its IP address, port number, and the total amount of shared files. To look for files shared by other network nodes, a query message is utilized. It has a query string and a minimum link speed requirement. One or more files matching a particular query are listed in a query-hit message, their sizes, and the responding node's connection speed. Push is used to upload files to clients protected by a firewall and unable to download files on their own.

Then, A and B communicate via the Gnutella protocol. A 23-byte descriptor with the following fields is included in every protocol message: "id, type, TTL, hops, payload length." A 16-byte descriptor number (approximately) unique on the network makes up the first field. In an overlay network, TTL represents the packet's time-to-live, and hops denote how many hops the message has taken. The hop count is modified each time a network node that the message passes through. The actual data in the Gnutella packet, if any, are described by the payload length. This value is significant since there are no gaps in the Gnutella datastream and several Gnutella packets may fit in a single IP datagram. Finally, the type specifies which of the five message types—Ping, Pong, Query, QueryHit, and Push—is contained in the packet.

4.5 Random forest for P2P categorization

This section provides a clear picture of the random forest algorithm and how proposed model works. In the previous section, as we talked about RFA (random forest algorithm), it can handle large amount of dataset, so there is a lot of flexibility in terms of how many trees can be run. There is no requirement for cross-validation in this approach. The basic classification framework employed is random forest. A random forest is made up of multiple tree-structured classifiers, as seen in Fig. 5. The foundation tree structure of the random forest in the proposed technique is decision tree and feature extraction. In the forest tree, all of the other trees have the same structure, and the bootstrap sample is generated using a random vector. Finally, the

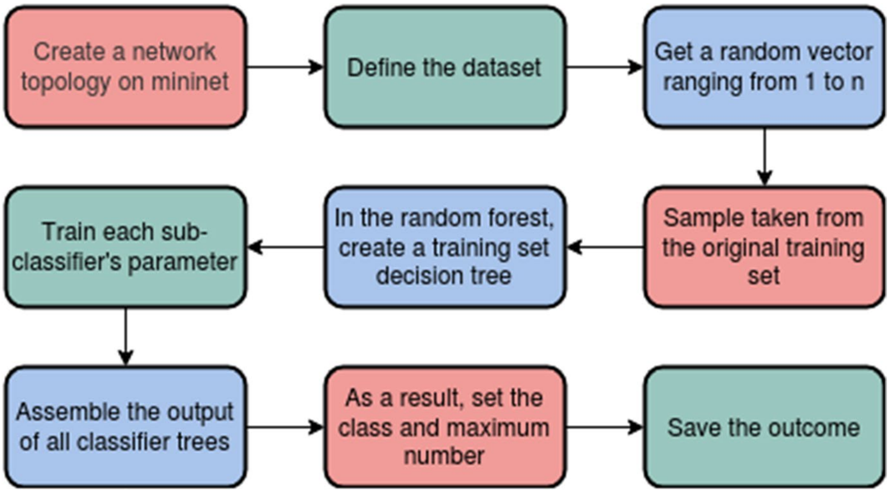
Create a network topology on mininet
Define the dataset
Get a random vector ranging from 1 to n
Sample taken from the original training set
In the random forest, create a training set decision tree
Train each sub-classifier's parameter
Assemble the output of all classifier trees
As a result, set the class and maximum number
Save the outcome

Fig. 7 Working of proposed solution

most popular class is chosen and categorized based on voting results. In our proposed solution, the main objective is feature ranking, selection and filtering of the preprocessing state using the random forest technique. The working of random forest algorithm is generalized as seen in Fig. 7.

By using this selection strategy, there may be a higher chance of choosing m better variables. Consequently, we may increase classification accuracy. The model will be built faster if the splitting variables are improved since each tree will grow faster. Another benefit of this approach is that, since a variable's significance to various applications varies, we could specify higher probabilities for such variables if, in particular circumstances, we are more interested in specific applications. As a result, we can improve these apps' categorization accuracy.

4.6 Random forest algorithm based on heuristic categorization

A supervised learning method known as "random forest" builds several decision trees throughout the training phase and outputs the class that represents the mean or average of all the classes. It is used in several applications, including regression and classification. The packet-based classification method receives the unclassified P2P data. On the basis of the suggested heuristic principles, traffic flows are categorized as P2P or non-P2P. If a flow of traffic meets any of the suggested criteria, it is categorized as a P2P flow, and the P2P flow database and destination IP table are modified appropriately. The packet-level categorization procedure is shown in Algorithm 1. P2P traffic is classified as the initial phase. The unclassified P2P traffic is sent into the flow-level classification mechanism.

Algorithm 1 Random Forest algorithm based on Heuristic categorization

```
Input: Packets of network traffic
Output: Categorization of P2P and non-P2P traffic flows
pac: packets
ftab: flow table of P2P
finfo: flow information of P2P
a: port number of source
b: port number of destination
c: IP table of destination
d: well known P2P ports
host1: Heuristic 1
host2: Heuristic 2
host3: Heuristic 3
host4: Heuristic 4

Start
1: pac = capture packet ()
2: do
3: {
4:      if (ftab.contains (pac.finfo)
5:              go to step 15
6:      else if (pac.a == d || pac.b == d)
7:      {
8:            write: pac.finfo →P2P
9:            update: c ←pac.finfo
10:     }
11:     else if ((pac.host1 || pac.host2 || pac.host3 || pac.host4) == true)
12:     write: pac.finfo →P2P
13:     else
14:     write pac.finfo →non P2P
15:     pac = capture packet ()
16: }
17: while (pac ! = null)
18: go to 2nd step for classification process
End
```

To aggregate the predictions of several decision trees into a single model, the main justification for using a random forest rather than a decision tree. According to this reasoning, a single model that is composed of several poor models would nevertheless be superior to a single excellent model. Given the general effectiveness of random forests, this is true. Because of this, random forests are less likely to overfit.

With a flexible model, such as decision trees, over-fitting might happen because the model will also remember the training data and learn any noise in the data. It won't be able to forecast the test findings as a result. By integrating several trees into a single ensemble model, a random forest helps minimize the high variance from a flexible model like a decision tree.

Random Forest is less costly to compute and does not need a GPU to complete training. A random forest may interpret a decision tree differently yet more effectively. For neural networks to be really successful, they will need a lot more data than the average human could have on hand. Simply said, the neural network will reduce your features' interpretability to the point where they are irrelevant for performance.

4.6.1 Flow diagram of proposed solution

The entire system of the P2P traffic categorization process is depicted in Fig. 8, which is separated into two steps, such as the packet-level process and the flow-level process. The P2P-port-based approach, in combination with the packet-heuristics-based method, conducts traffic categorization at the packet level. The traffic not categorized as P2P in the first stage is put into the flow-level classification stage, which uses flow-heuristics in conjunction with a statistical-based approach to classify the remaining data. Mininet simulator is used to implement the suggested approach on Ubuntu Linux.

A collection of five network factors (source-IP, destination-IP, source-port, destination-port, and protocol) are utilized to define traffic flow when categorizing traffic. Any communication between the two processes would share these five factors. Packets related to the same flows are identified by computing the hash-key of the packet by integrating the five-tuple flow information in the packet-level classification procedure. The hash key would be the same for packets belonging to the same flow traveling in

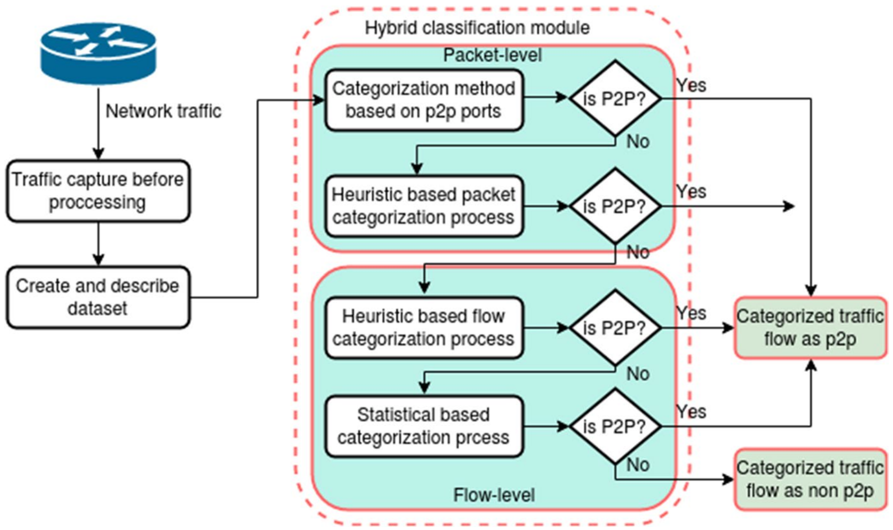
Network traffic
Traffic capture before proccessing
Create and describe dataset
Hybrid classification module
Packet-level
Categorization method based on p2p ports
is P2P?
Yes
No
Heuristic based packet categorization process
is P2P?
Yes
No
Heuristic based flow categorization process
is P2P?
Yes
No
Statistical based categorization prcess
is P2P?
Yes
No
Flow-level
Categorized traffic flow as p2p
Categorized traffic flow as non p2p

Fig. 8 Flow diagram of proposed solution

any direction. This hash-key is important for determining whether or not the packets in the flow have already been classed as P2P or non-P2P.

5 Performance Evaluation and implementation

In order to test the performance and efficiency of the proposed system, the simulation environment was chosen to consist of famous networking tools and devices. For this purpose, Ubunto 20.04 LTS operating system, Mininet simulator (a software program to simulate the network in a laptop), POX SDN Controller, OpenFlow switches and routers, topology generator, etc., are used. Firstly, the topology was created using this specific tool and integrated with the SDN controller. Then, multiple SDN switches, hosts, and a controller were added to create a network in Mininet according to the scenario according to the problem statement and projected solution. Finally, the performance was evaluated in terms of bandwidth usage, propagation delay, success rate query, average response time, P2P computation overhead, overall dropped packets, etc.

5.1 Evaluated parameters

The following parameters were used to validate the performance of the proposed methods, which is under discussion: (1) bandwidth usage, (2) propagation delay, (3) success rate query, (4) average response time, (5) P2P computation overhead, and (6) overall dropped packets. The parameters are defined as follows:

5.1.1 Network bandwidth usage

The greatest amount of data that may be sent across a wired or wireless communications link in a certain amount of time is referred to as P2P network bandwidth consumption. The amount of bits, kilobits, megabits, or gigabits that may be transferred in a second is a common way to describe bandwidth.

5.1.2 Propagation delay

The travel time of packets through the transmission channel is described as propagation delay and is limited by the speed of light.

$$\text{Tp} = \text{Distance/Velocity}$$

If the medium's distance is greater, it takes longer to arrive at the desired location. And packets will be delivered faster if the signal's velocity (speed) is high.

5.1.3 Success rate query

The success rate is a critical indicator for assessing the efficacy of the P2P resource finding method. It is a proportion of the total number of created requests to the total number of successfully received answers. It is expressed as a percentage as follows:

$$\text{Success rate} = \text{Rep}/\text{Req} \ * 100$$

where Req is the total number of resource requests issued and Rep is the total number of correctly received resource replies.

5.1.4 Average response time

Another key statistic for assessing the efficacy of the P2P resource discovery method is response time. It is the time differential between the resource request's production and the resource response's receipt. It is calculated in the following manner:

$$\text{Average response time} = T_{\text{Rep}} - T_{\text{Req}}(\text{s})$$

5.1.5 P2P computation overhead

This paper defines Overhead as the total amount of routing messages created and sent at the network layer.

5.1.6 Overall dropped packets

Packet drop is an essential indicator for assessing network performance. In a wireless Ad hoc network environment, packets will be discarded for a variety of reasons. Dropping of packets is projected to be significant in a typical P2P network since these techniques would employ many repetitions of the same resource discovery message over the whole resource discovery process, increasing overhead and generating network bottlenecks. The number of packets dropped at each network layer of all P2P nodes were counted.

5.2 Results and discussion

This suggested approach is compared to two fundamental protocols: flooding and random walk. A good resource discovery protocol uses less bandwidth (i.e., fewer throughput) and has an equal success rate. Figure 9 depicts throughput comparisons for several procedures. Although the present technique uses less bandwidth, there is a compromise in terms of success rate. This proposed technique outperforms the others when compared. Based on the simulation output shown in Fig. 9, all end nodes are receiving a significant network throughput. From the graph, the proposed solution has received priority access to the network capacity as observed. This indicates that the bandwidth reservation carried out following the SDN Controller's instructions received the desired and consistent receiving throughput.

Figure 10 depicts the measurement of propagation delay to monitor the effectiveness of the P2P protocol. Only correctly transmitted data packets are examined for each resource discovery step. The random walk has a low performance. This recommended technique yields positive outcomes.

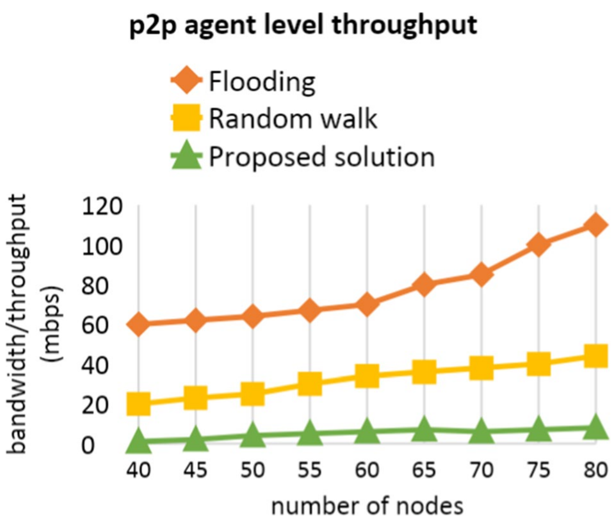

Fig. 9 Network bandwidth usage

Fig. 10 P2P network propagation delay

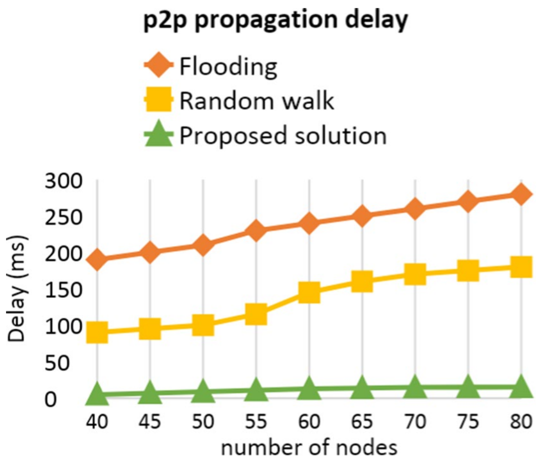

Figure 11 depicts the successful inquiries as a ratio of the total produced requests to successful answers during the course of the experiment. This proposed strategy outperforms the flooding method in terms of performance. The random walk strategy is ineffective. The success rate of each technique declines as bandwidth use rises. The cause of this is that when bandwidth use rises, network congestion rises as well. More defects result from increased congestion, which lowers the success rate. Flooding has the lowest average success rate because it lacks a link load balance strategy, resulting in the biggest number of errors. While the proposed solution has a link load balance mechanism and robust contracts, which results in a higher average success rate.

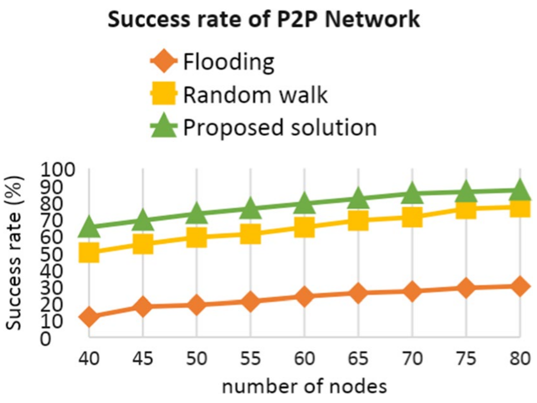

Fig. 11 Success rate query

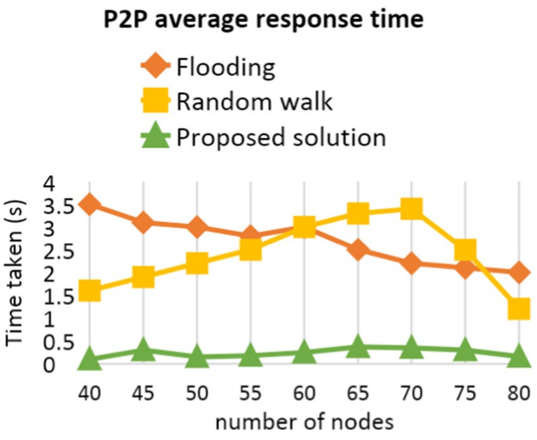

Fig. 12 Average response time

The query response time is utilized as a measure in Fig. 12 to validate the performance of the resource discovery technique. The query response time is defined as the shortest time difference required by the protocol to resolve the question. The algorithm with the shortest query latency is the best technique. This proposed technique outperforms the current methods.

Figure 13 compares network overhead for our proposed and existing strategies. The overhead is calculated in terms of routing packets produced. When compared to competing protocols, this protocol has relatively minimal overhead. This is due to the usage of conditional addressing. The overhead in the random walk protocol is substantial owing to unicast message delivery.

Figure 14 shows the results of the lost packet count. The total packet lost statistic is calculated as the difference between the number of packets sent and the number of packets received. The total number of packets lost throughout the simulation is counted in this example. The graph illustrates that suggested approach outperforms all current resource discovery algorithms. Because of the message forwarding architecture, which has no effect on the resource discovery process, this suggested protocol performs better.

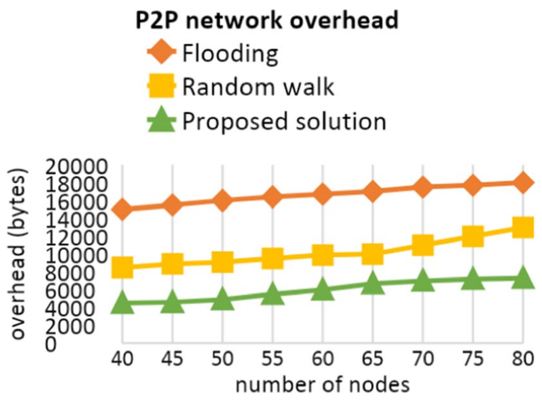

Fig. 13 P2P network generated overhead

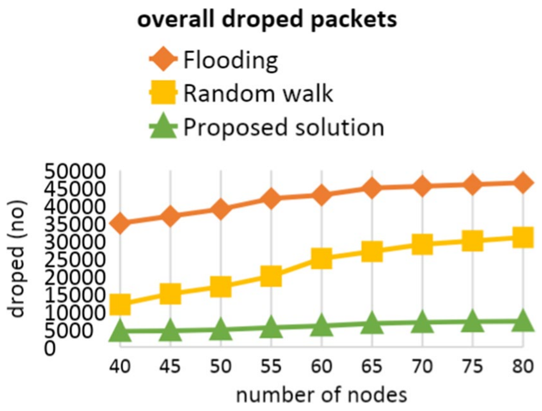

Fig. 14 Dropped packets count

5.3 Computational cost

The space complexity of a program determines how much memory it needs to function. Because a program needs memory to store input data and temporal values while it is running, space complexity is auxiliary and input space. Similar to how temporal complexity aids in solution evaluation, it does the same. For a Random Forest of size F and maximum depth L (excluding the root), the computational complexity at test time is O(F.L). However, the computational cost may be cheaper as trees are not balanced. In the regime where Random Forest works, the unification costs of ensemble techniques, which were neglected in the theoretical cost calculation since they are often insignificant, become extremely substantial and account for the majority of the overall cost. Memory space, which is exponential in the tree's depth and must be considered while using our technique, has the formula O(2L).

6 Conclusion

To simplify bandwidth management in peer-to-peer networks, this research paper has suggested a paradigm based on the SDN idea. According to the kind of traffic each sender in the network carries, this suggested model makes it simple to reserve a certain amount of network bandwidth for them. Because of the dynamic nature of its nodes' joining and leaving, a Peer-to-Peer (P2P) streaming network is typically imbalanced and suffers from delivery delays. As a result, dynamic network reconfiguration is necessary to maintain the network balanced. An SDN-based adaptive reconfiguration of the structure and routing in a P2P streaming network is proposed in this research. This technique allows for entirely decentralized data distribution under centralized supervision, allowing for optimal bandwidth management through a machine learning algorithm. A unique approach is suggested that uses the Random Forest algorithm to regulate connection bandwidth based on the network's needs. In comparison to existing techniques, these simulations suggest that the proposed solution has achieved a good performance.

Acknowledgements The research reported in this publication was supported by the Deanship of Scientific Research and Innovation at Al-Balqa Applied University in Jordan. Grant Number: DSR-2021#382.

Data Availability Data may be available from the corresponding author on request.

Declaration

Conflict of interest Author declares no conflict of interest.

References

1. Bilal R, Bilal MK (2019) Software-defined networks (SDN): a survey. In Handbook of research on cloud computing and big data applications in IoT, pp. 516–536. IGI Global.
2. Amin R, Reisslein M, Shah N (2018) Hybrid SDN networks: a survey of existing approaches. IEEE Commun Surv Tutorials 20(4):3259–3306
3. Shirmarz A, Ghaffari A (2020) Performance issues and solutions in SDN-based data center: a survey. J Supercomput 76(10):7545–7593
4. Xia W, Di C, Guo H, Li S (2019) Reinforcement learning based stochastic shortest path finding in wireless sensor networks. IEEE Access 7:157807–157817
5. Hussain M, Nadir S, Rashid A, Sultan SA, Aziz A, Syed MR (2022) Software-defined networking: categories, analysis, and future directions. Sensors 22(15):5551.
6. Touqeer H, Zaman S, Amin R, Hussain M, Al-Turjman F, Bilal M (2021) Smart home security: challenges, issues and solutions at different IoT layers. J Supercomput, pp.1–37.
7. McCauley J, Liu Z, Panda A, Koponen T, Raghavan B, Rexford J, Shenker S (2016) Recursive SDN for carrier networks. ACM SIGCOMM Comput Commun Rev 46(4):1–7
8. Yoichi R, Sugawara S (2017) Consistency preservation of replicas based on access frequency for content sharing in hybrid peer-to-peer networks. Int J Space-Based Situated Comput 7(4):197–206
9. Yeganeh SH, Amin T, Yashar G (2013) On the scalability of software-defined networking. IEEE Commun Magaz 51.2:136–141.
10. He Y, Richard YF, Nan Z, Victor CML, Hongxi Y (2017) Software-defined networks with mobile edge computing and caching for smart cities: a big data deep reinforcement learning approach. IEEE Commun Magaz 55(12):31–37.

11. Hu Z et al (2015) A comprehensive security architecture for SDN. In: Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on IEEE, New York.
12. Casado M, Freedman MJ, Pettit J, Luo J, McKeown N, Shenker S (2007) Ethane: taking control of the enterprise. In: Proc. Conf. Appl. Technol. Architect. Protocols Comput. Commun., pp. 1–12.
13. Greenberg A, Hjalmtysson G, Maltz DA, Myers A, Rexford J, Xie G, Yan H, Zhan J, Zhang H (2005) A clean slate 4D approach to network control and management. ACM SIGCOMM Comput Commun Rev 35(5):41–54
14. Casado M, Garfinkel T, Akella A, Freedman MJ, Boneh D, McKeown N, Shenker S (2006) SANE: a protection architecture for enterprise networks. In: USENIX Security Symposium 49:50
15. Faigl Z, Zsolt S, Robert S (2014) Application-layer traffic optimization in software-defined mobile networks: a proof-of-concept implementation. In: 2014 16th International Telecommunications Network Strategy and Planning Symposium (Networks), pp. 1–6. IEEE, New York.
16. Nunes, BA et al (2014) A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun Surv Tutorials 16.3 :1617–1634.
17. Kobo, H.I., Abu-Mahfouz, A.M. and Hancke, G.P., 2018. Fragmentation-based distributed control system for software-defined wireless sensor networks. IEEE transactions on industrial informatics, 15(2), pp.901–910.
18. Khondoker R et al (2014) Feature-based comparison and selection of Software Defined Networking (SDN) controllers. In: Computer Applications and Information Systems (WCCAIS), 2014 World Congress on. IEEE, New York.
19. Vicino Damián C-H, Lung GW, Olivier D (2015) Investigation on software-defined networks' reactive routing against BitTorrent. IET Netw 4(5):249–254.
20. Lin Thomas et al (2014) Enabling Sdn applications on software-defined infrastructure. In: Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, New York.
21. Amin R, Bilal AM, Aftab AK (2010) Analyzing performance of ad hoc network mobility models in a peer-to-peer network application over mobile ad hoc network. In: 2010 International Conference on Electronics and Information Engineering, vol. 2, pp. V2–344. IEEE, New York.
22. Li Y et al (2015) A unified control and optimization framework for dynamical service chaining in software-defined NFV system. IEEE Wireless Commun 22.6:15–23.
23. Choi T, Kang S, Yoon S, Yang S, Song S, Park H (2014) "SuVMF: Software-defined unified virtual monitoring function for SDN-based large-scale networks." In Proceedings of The Ninth International Conference on Future Internet Technologies, pp. 1–6.
24. Fotiou N et al (2015) H-pastry: an inter-domain topology aware overlay for the support of name-resolution services in the future internet. Computer Commun 62:13–22.
25. Liu Z, Jie Z, Jiangmei Z, Qingli L (2019) Routing algorithm design of satellite network architecture based on SDN and ICN. Int J Satellite Commun Netw (2019).
26. Zarca AM, Dan G-C, Jorge BB, Jordi O, Rafael M-P, Antonio S (2019) Enabling virtual AAA management in SDN-based IoT networks. Sensors 19(2):295.
27. Yeh C-C, Qiu J-W, Chang S-J (201) SEAL2: An SDN-enabled all-Layer2 packet forwarding network architecture for multitenant datacenter networks. Int J Commun Syst, e3937.
28. Harun A, Mohamed RAR, Ruslan AR, Lili EMR, Musaddiq MK, Amina SAB (2018) The emotional impact of humanized brand: case study of Malaysian SME food products. Adv Sci Lett 24(4):2451–2454.
29. Verleysen E, Van Doren E, Waegeneers N, De Temmerman P-J, Abi Daoud Francisco M, Mast J (2015) TEM and SP-ICP-MS analysis of the release of silver nanoparticles from decoration of pastry. J Agric Food Chem 63(13):3570–3578.
30. Hwang I, Rianto A, Pakpahan AF (2018) Software-defined Peer-to-Peer file sharing architecture for TWDM PON. In: 27th Wireless and Optical Communication Conference (WOCC). Hualien, pp 1–4. https://doi.org/10.1109/WOCC.2018.8372713
31. Tran HM, Son TL, Hai-Duong L, An TT (2018) A distributed controller approach using P2P protocol for software defined networks. In: 2018 International Conference on Advanced Computing and Applications (ACOMP), pp. 65–70. IEEE, New York.
32. Struhár V, Ashjaei M, Behnam M, Craciunas SS, Papadopoulos AV (2019) DART: Dynamic bandwidth distribution framework for virtualized software defined networks. In: Proceedings of the 45th Annual Conference of the IEEE Industrial Electronics Society (IECON), Lisbon, Portugal, 14–17 October 2019; Volume 1, pp. 2934–2939.

33. Chang Y, Chen Y, Chen T, Chen J, Chiu S, Chang W (2019) Software-defined dynamic bandwidth management. In: Proceedings of the 2019 21st International Conference on Advanced Communication Technology (ICACT), PyeongChang, Korea, 17–20 February 2019, pp. 201–205.
34. Jimson ER, Nisar K, bin Ahmad Hijazi MH (2017) Bandwidth management using software defined network and comparison of the throughput performance with traditional network. In: Proceedings of the International Conference on Computer and Drone Applications (IConDA), Kuching, Malaysia, 9–11 November 2017; pp 71–76.
35. Javadpour A, Wang G (2022) cTMvSDN: improving resource management using combination of Markov-process and TDMA in software-defined networking. J Supercomput 78:3477–3499. https://doi.org/10.1007/s11227-021-03871-9

Authors and Affiliations

Hamza Aldabbas[1]

[1] Prince Abdullah Bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University, Al-Salt, Jordan