# Dynamic Traffic Engineering for Cooperative Fog-Cloud Environment: Trade-off Analysis of Cost and Utilization Under Different Load Conditions

Md. Rahinur Rahman, and Mirza Mohd Shahriar Maswood

***Abstract*—Fog computing has become an attractive computing method for different IoT (Internet of Things) applications that require low latency and location awareness. It provides low latency by bringing computational power closer to the network edge, working as a complement to cloud computing. Despite its advantages, fog computing faces challenges due to the limited resources (CPU processing capacity, network bandwidth, memory, and power backup) of fog nodes. This work introduces a novel optimization model for a cooperative fog-cloud environment dealing with dynamic traffic. We analyze how different arrival rates impact bandwidth costs, link utilization, and server resource utilization. Our results show that fog resource utilization is greater than cloud resource utilization under varying traffic conditions, with blocking rates remaining within an acceptable range (0-15%). The key contributions include the formulation of an optimization model that optimizes resource allocation, addresses blocking factors in fog networks, and offers valuable insights for managing dynamic traffic in fog computing networks.**

***Index Terms*—Fog Computing, Dynamic Traffic Engineering, IoT, Resource Utilization, Blocking Rates.**

## I. Introduction

In the current era of overwhelming Internet access, the Internet of Things (IoT) has become an important part of our daily lives. It enables seamless interaction between devices and humans, without the need for manual intervention. IoT is demonstrating its potential in home automation, smart cities, industrial automation, agriculture, healthcare, education, transportation systems, and various other smart applications [1], [2]. With time, the number of connected IoT devices is increasing, and the number is projected to be 500 billion by 2030 [3], [4]. A white paper by Cisco Systems Inc. reveals that intranet and Internet traffic is doubling every 100 days [5].

Conventionally, IoT traffic is routed to the remote cloud for processing, and after processing, the result is sent back to the edge or stored at the cloud. This conventional technique is considered costly and time-consuming. Additionally, cloud computing suffers from a lack of mobility support, location awareness, and sometimes, a concern for privacy [6]. For latency-sensitive applications, it is necessary to process IoT traffic closer to avoid long-distance communication among IoT devices and the cloud. This can be achieved through fog computing, which works in conjunction with cloud computing [7]. Fog nodes provide the data processing facility in a decentralized way. Fog nodes are low-to-moderately resourceful devices such as set-top boxes, smart gateways, switches, routers, and base stations, and resourceful appliances like cloudlets or dedicated fog devices [8], [9]. Collaboration among these fog nodes plays a vital role in making the fog computing paradigm more successful. For the effective utilization of fog resources, server-level and link-level load balancing is important. Cooperative fog computing can provide better load distribution and avoid over and under-utilization of fog nodes.

Our goal is to adopt the fog cloud computing model to facilitate novel applications of fog computing, such as real-time tracking of vehicles and patients, smart homes, smart cities, smart agriculture, and more. According to fog We formulated our model as an abstraction of fog layer-1, fog layer-2, and cloud layers with realistic resource capacities to give a clear idea to fog service providers. We have kept the provision for fog service providers to introduce dedicated fog analytics devices in this model, which would facilitate deploying real fog computing services.

The main contributions of our works are as follows:

• Development of a Mixed Integer Linear Programming (MILP)-based optimization model for dynamic traffic in fog-cloud environments.

• Comprehensive analysis of fog and cloud layer resource utilization (bandwidth, link, and server capacity) under varying traffic loads.

• Introduction of blocking rate analysis to evaluate system performance under resource constraints, which has not been explored for fog computing in the existing literature.

The rest of this paper is organized as follows: Section II reviews related works on fog computing applications, optimization models, and cooperative traffic management. Section III covers system assumptions, problem formulation, and the cooperative fog computing architecture. Section IV discusses simulation procedures and analyzes results on bandwidth costs, resource utilization, and blocking rates. Section V concludes this work and outlines our future research directions.

Manuscript received August 26, 2024; revised September 10, 2024. Date of publication October 10, 2024. Date of current version October 10, 2024. The associate editor prof. Claudia Canali has been coordinating the review of this manuscript and approved it for publication.

Authors are with the KUET, Bangladesh (e-mails: shahriar@ece.kuet.ac.bd, rahinur2k11@gmail.com.)

Digital Object Identifier (DOI): 10.24138/jcomss-2024-0061

## II. Related Work

Researchers from various fields, including Cloud Computing (CC), Mobile Cloud Computing (MCC), Mobile Edge Computing (MEC), Content Delivery Network (CDN), and IoT, are advancing fog computing through new architectures and frameworks to optimize fog resource utilization [10]. In [11], authors propose novel applications of fog computing for healthcare systems, where fog computing is employed for on-site analysis of wearable devices and environmental sensor data. Fog computing shows significant potential for multimedia applications by enabling low-latency, high-throughput experiences [12], [13]. Additionally, it plays a crucial role in big data processing within smart grids, facilitating tasks such as real-time monitoring and load balancing [14].

For fog computing systems, data acquisition and its modeling are important [15]. In [16], authors demonstrated that the nature of aggregated IoT traffic can be modeled as a Poisson distribution. In [17], authors proposed constrained programming to solve the service placement problem, where application arrivals follow a Poisson distribution. In [18], authors discuss how traffic patterns in fog networks are constantly changing, i.e., the dynamic nature of traffic, and suggest an algorithm for placing virtual machines optimally to reduce latency and maximize resource use. In [19], the author developed a dynamic traffic engineering model with linear programming to improve network resource utilization for integrated services networks. The dynamic traffic engineering method is used in [20] for maintaining effective quality of service (QoS) for software-defined overlay networks. In the case of QoS improvement for dynamic traffic engineering, resource shortages are common, so identifying the type of shortage is necessary.

Depending on the nature of IoT traffic and computing effectiveness, authors are proposing cooperative and non-cooperative fog computing frameworks. Cooperative fog computing is a powerful computing paradigm for resource-constrained environments [21]. To handle dynamic traffic from different regions, cooperative fog computing is more effective than non-cooperative fog computing. To address the performance of any fog computing architecture, fog-to-fog collaboration is desired [22]. Cooperative fog computing has the ability to maximize network throughput and efficiently utilize resources [23], [24]. Task offloading in the case of fog computing uses different approaches depending on the collaboration strategy. In [25], authors proposed a hierarchical architecture for vehicular fog computing and utilized a task offloading strategy to improve computing efficiency.

Balancing loads in cooperative fog computing is also vital for avoiding overutilization of any servers or links [26]. Researchers use a variety of load-balancing techniques to increase service quality and resource utilization. For continuous monitoring of fog computing nodes, links, and allocation of resources for incoming requests in an efficient manner, a central monitoring system is required [26]. Software or hardware equipment is utilized as a manager to distribute loads among various resourceful nodes [27]. Also, consideration of fog resources for serving as the central controller for balancing loads in any fog computing system is beneficial in terms of capital cost. Balancing loads in edge computing is analyzed in [28]; they use intermediary nodes as a central controller. In [29], authors proposed a dynamic load balancing technique considering the dynamic nature of cooperative fog computing systems. As there are more and more fog nodes' involvement in this processing, server-level load balancing could speed up the processing scheme.

Also, bandwidth cost, delay, and energy cost minimization for cooperative fog computing have become attractive research topics in recent times for making this computing suitable for IoT applications [30], [31]. In [3] and [32], authors accomplished delay and energy minimization for edge computing and fog computing. In [33], authors proposed an efficient scheduling algorithm to minimize delay and energy cost in fog computing. Their scheduling algorithm performed well to reduce delay and energy consumption compared to the non-preemptive algorithm FCFS. The task scheduling problem also gets attention in [34]. In this work, authors propose modified artificial ecosystem-based optimization in fog cloud environments for effective IoT task scheduling. Their alternative task scheduling algorithm improved the system's performance. Also, in [35] authors have utilized fog computing for smart farming with energy, delay, and resource consumption analysis. In our previous work [36] we accomplished bandwidth cost minimization, link-level load balancing, and server-level load balancing for three layer fog cloud computing.

Maintaining Quality of Service (QoS) for fog computing is another significant challenge [37]. Fog service disruption due to resource unavailability can occur, especially during periods of burst traffic. The vehicular network environment poses traffic congestion due to the bursty nature of vehicular traffic data [38]. In this work, authors propose a clustering-based computation method for reducing cloud traffic on the network. In [39], authors proposed a model for managing local traffic in vehicular fog computing and demonstrated traffic congestion mitigation through simulation. Due to the heavy involvement of local nodes, effective resource-level load balancing and service blocking analysis are essential. Service occlusion in geo-distributed cloud data centers is discussed in [40], [41]. Fog computing faces greater resource shortages than diversified data centers, necessitating a robust framework for maintaining service availability. Studies [42], [43] covered service availability and reliability, and [44] proposed an optimization model for reliability and QoS, focusing on reducing bandwidth waste and energy consumption. However, the impact of server and network resource shortages on user requests remains unexplored.

So far works we discussed here, no one analyzed how fog computing resource usage varies for different arrival rates of requests. In addition to analyzing a few metrics such as bandwidth cost (cost-effectiveness), link utilization, and server resource utilization, we also conducted an in-depth analysis of the blocking rate due to the non-availability of fog resources in fog computing environments. Notably, the analysis of blocking was not explored in the existing literature we reviewed. This is the key contribution of our work beyond the state-of-the-art progress, and thus our work fills a significant gap in this

TABLE I
COMPARISON OF RELATED WORKS FROM DIFFERENT PERSPECTIVES

| Parameters<br>Reference Papers | Optimization type | Impact analysis of arrival rate | Extraction of layer-wise resource utilization | Analysis of blocking rate | Perception of service disruption factors |
|---|---|---|---|---|---|
| [30] SP Singh et al. | Energy cost and delay | No | No | No | No |
| [45] G Shruthi et al. | Server resource and execution time | No | No | No | No |
| [46] LA Phan et al. | Link and server resource | No | No | No | No |
| [31] Kadhim et al. | Network, server resource, and Delay, | No | No | No | No |
| [47] S Bebortta et al. | Energy cost and delay | No | No | No | No |
| [48] Mukae et al. | Energy cost and delay | No | No | No | No |
| [49] Saif et al. | Energy cost and delay | No | No | No | No |
| [50] Singhal et al. | Energy cost and delay | No | No | No | No |
| [51] MB Kamal et al. | Server resource | No | No | No | No |
| [52] N Villegas et al. | Energy and monetary cost | Yes | No | No | No |
| [36] MMS Maswood et al. | Bandwidth cost, link/network, and server resource | No | No | No | No |
| Our proposed | Bandwidth cost, link/network, and server resource | Yes | Yes | Yes | Yes |

area of research. Also, we believe our optimization model will enable fog service providers to assess the fog architecture's responsiveness to dynamic traffic patterns.

## III. SYSTEM ASSUMPTION AND PROBLEM FORMULATION

Figure 1 shows the fog computing architecture. According to [53] and [36], while assuming a fog computing system for the better utilization of fog resources, we considered three layers, namely fog layer-1, fog layer-2, and cloud. Fog layer-1 has less powerful computing resources than fog layer-2 and cloud. According to [28] bandwidth of fog layer-1 is broader than fog layer-2 and cloud. Given their proximity, we assume the bandwidth cost of fog layer-1 is lower than that of fog layer-2 and the cloud. The fog layer-1 is divided into four regions, and in each region, there are many IoT devices. Each region has a central gateway called a cluster point (CP). Sensor nodes send their data to their regional CP, which needs further processing and is denoted as a request. Each request consists of two tuples: $\langle h, r\rangle$ where $h$ is the bandwidth demand and $r$ is the server resource demand. Demands are aggregated at CP, forming as $H_i$ aggregated bandwidth demand and $R_i$ as aggregated resource demand. In our framework, we model demand generation using a Poisson distribution, and arrival rates influence the duration of demand.

In this work, we assume that cost-effective fog servers in a CP's own region primarily handle the requests. If the region's servers cannot meet the demand, the request is forwarded to other regions within fog layer-1. If the fog layer-1 servers' resources are not sufficient, then the request is placed at fog layer-2 and finally at the cloud, which is the most costly layer. A Software Defined Network (SDN) controller will be used to solve our formulated MILP optimization model and allocate resources in real time based on demand. The controller can either be hosted on a dedicated server by fog service providers or on layer-2 fog computing nodes with sufficient resources. When any CPs have aggregated demand, first the CP's aggregated demand profile (which is a few KB in size and can be neglected in bandwidth consumption) is sent to the SDN controller. The SDN controller then decides suitable servers and sends the data forwarding rules to the CP. The CP updates its data forwarding table and sends requests accordingly.

The meaning of the notations used to formulate the model is explained in Table II.

### A. Constraints

We consider $H_i$ as the aggregate bandwidth demand generated from all sensor nodes in the region of cluster point $i$. Therefore, $H_i$ can be expressed mathematically as:

$$H_i(t) = \sum_{n \in N_i} h_{ni}(t), \quad i \in I \tag{1}$$

Similarly, we denote the total resource demand generated by all sensor nodes under cluster point $i$ by $R_i$.

$$R_i(t) = \sum_{n \in N_i} r_{ni}(t), \quad i \in I \tag{2}$$

The generated bandwidth demand from each zone $i$ can be satisfied by the available bandwidth of its own region, other regions of fog layer 1, fog layer 2, or the cloud. It may appear

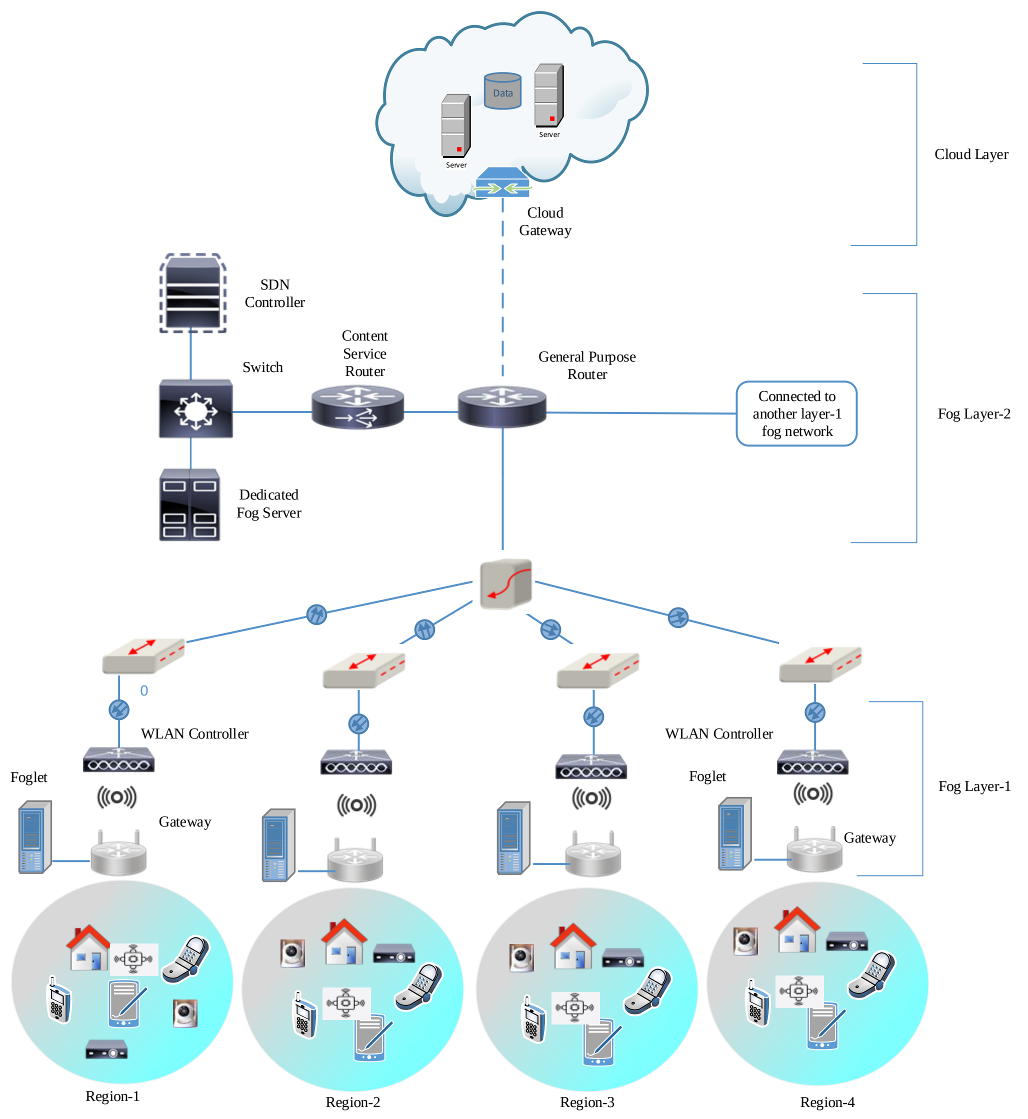

Fig. 1. Fog computing architecture.

that at any review point, requests can experience blocking due to a shortage of bandwidth resources to reach a server. In this case, an artificial bandwidth resource allocation $\widetilde{y}_i(t)$ is considered.

$$\widetilde{y}_i(t) + \sum_{s \in S} y_i^s(t) = H_i(t), \quad i \in I \tag{3}$$

A binary variable $\widetilde{f}_i(t)$ is considered to indicate the artificial allocation of bandwidth resources.

$$\widetilde{y}_i(t) \leq M\widetilde{f}_i(t), \quad i \in I \tag{4}$$

Additionally, we can serve a request from a cluster point using either real or artificial allocation of bandwidth resources, but these two options are mutually exclusive.

$$\widetilde{f}_i(t) + q_i(t) = 1, \quad i \in I \tag{5}$$

For a real allocation of bandwidth resources, the total amount of links' bandwidth required to serve a request can be expressed as:

$$\sum_{s \in S} y_i^s(t) = H_i(t)q_i(t), \quad i \in I \tag{6}$$

Path flow variables are used to establish paths from the source cluster points to destination servers. The bandwidth demand allocated over the path from cluster point $i$ to destination server $s$ is denoted as $x_{ip}^s$:

$$\sum_{p \in P_{ip}^s} x_{ip}^s(t) = y_i^s(t), \quad i \in I, s \in S \tag{7}$$

Links that form a path must carry the fraction of demand $H_i$ if any bandwidth is allotted to that path $p$. Therefore, by

TABLE II
NOTATIONS USED IN FORMULATION

| Constants: |
|---|
| $S$ = Set of servers. |
| $I$ = Set of cluster points (CP). |
| $N_i$ = Set of sensor nodes under cluster point $i$. |
| $L$ = Set of links. |
| $P_{ip}^{s}(t)$ = Set of paths from CP $i$ to server $s$ at review point $t$. |
| $h_{ni}(t)$ = Bandwidth demand generated by sensor node $n$ at CP $i$ at review point $t$. |
| $r_{ni}(t)$ = Resource demand generated by sensor node $n$ from CP $i$ at review point $t$. |
| $H_i(t)$ = Aggregated bandwidth demand generated by all sensor nodes from CP $i$ at review point $t$. |
| $R_i(t)$ = Aggregated Resource demand generated by all sensor nodes from CP $i$ at review point $t$. |
| $c_l(t)$ = Available capacity on link $l$ at review point $t$. |
| $a_s(t)$ = Capacity of server $s$ at review point $t$. |
| $\delta_{ipl}^{s}(t)$ = Link-path indicator: 1 if path $p$ which is set up from CP $i$ to server $s$ uses link $l$ in order to satisfy demand of CP $i$ by server $s$ at review point $t$, 0 otherwise. |
| $\alpha, \beta, \gamma$ = Weight parameters related to three optimization objectives. |
| **Variables:** |
| $y_i^s(t)$ = Bandwidth allocation for traffic from CP $i$ to server $s$ at review point $t$. |
| $\widetilde{y}_i(t)$ = Artificial bandwidth allocation for traffic from CP $i$ if request can not be served due to bandwidth shortage at review point $t$. |
| $\widetilde{f}_i(t)$ =Binary variable to indicate artificial bandwidth allocation. |
| $q_i(t)$ =Binary variable to indicate real bandwidth allocation. |
| $x_{ip}^{s}(t)$ = Bandwidth allocation in path $p$, if traffic from CP $i$ to server $s$ uses path $p$ at review point $t$. |
| $z_{il}^{s}(t)$ = Total bandwidth demand from link $l$ for requests generated from CP $i$ and served by server $s$ at review point $t$. |
| $u(t)$ = Maximum utilization of all links at review point $t$. |
| $g_i^s(t)$ = Resource allocation for traffic from CP $i$ to server $s$ at review point $t$. |
| $\widetilde{g}_i(t)$ = Artificial bandwidth allocation for traffic from CP $i$ if request can not be served due to server resource shortage at review point $t$. |
| $\widetilde{j}_i(t)$ = Binary variable to indicate artificial server resource allocation. |
| $d_i(t)$ =Binary variable to indicate real server resource allocation. |
| $\widetilde{b}_i(t)$ = A binary variable to indicate artificial allocation for both bandwidth and server resource shortages. |
| $k(t)$ = Maximum utilization of all servers at review point $t$. |

considering every path from $i$ to $s$, we can determine the flow on each link $l$:

$$\sum_{p \in P_{ip}^{s}} \delta_{ipl}^{s}(t) x_{ip}^{s}(t) = z_{il}^{s}(t), \quad l \in L, i \in I, s \in S \tag{8}$$

We need to assign another constraint to maintain link-level load balancing and prevent overuse of any links. The whole bandwidth allocated from any link cannot exceed the maximum link utilization times its capacity.

$$\sum_{i \in I} \sum_{s \in S} z_{il}^{s}(t) \le c_l(t) u(t), \quad l \in L \tag{9}$$

Note that the maximum utilization of any link cannot be greater than 1 at any point.

$$0 \le u(t) \le 1. \tag{10}$$

Similar to Equation 11, we satisfy the resource demand generated from region $i$ using the available resources from its own region, other regions in fog layer-1, fog layer-2, or the cloud. Similar to bandwidth shortages, situations of server resource shortages may occur. In the case of server resource shortages, we consider an artificial allocation of server resources denoted as $\widetilde{g}_i(t)$.

$$\widetilde{g}_i(t) + \sum_{s \in S} g_i^s(t) = R_i(t), \quad i \in I \tag{11}$$

To keep track of the artificial allocation of server resource shortages, a binary variable $\widetilde{j}_i(t)$ is also considered.

$$\widetilde{g}_i(t) \le M \widetilde{j}_i(t), \quad i \in I \tag{12}$$

As real and artificial allocation of server resources for a request is mutually exclusive, we can express it as eqn. 5.

$$\widetilde{j}_i(t) + d_i(t) = 1, \quad i \in I \tag{13}$$

So, for the real allocation of server resources, the total server resource allocation can be expressed as:

$$\sum_{s \in S} g_i^s(t) = R_i(t) d_i(t), \quad i \in I \tag{14}$$

We use constraint (15) to maintain proportionality between bandwidth allocation and server resource allocation. If more bandwidth is allocated to a path from cluster point $i$ to server $s$, then more computational capacity will be used from that server $s$.

$$H_i(t) g_i^s(t) = R_i y_i^s(t), \quad i \in I, s \in S \tag{15}$$

Similar to constraint (9) to maintain server-level load balancing and prevent overutilization of any servers, another constraint is required. The total amount of server resource allocation from any server cannot exceed the maximum server utilization times its capacity.

$$\sum_{i \in I} g_i^s(t) \le a_s(t) k(t), \quad s \in S \tag{16}$$

Since the maximum utilization of any server $s$ cannot exceed 1.

$$0 \le k(t) \le 1 \tag{17}$$

To keep track of blocking that happens in the system due to a shortage of bandwidth and server resources, we introduce another binary variable, $\widetilde{b}_i(t)$ as:

$$\widetilde{b}_i(t) = \widetilde{f}_i(t) \widetilde{j}_i(t), \quad i \in I \tag{18}$$

## *B. Objective Function*

There are three goals of this work: minimizing the bandwidth cost of routing, maximizing link utilization, and maximizing server resource utilization. These goals are presented with the composite objective function as given in eqn. 19. By varying the three weight factors $\alpha$, $\beta$, and $\gamma$ we can adjust the priority associated with each part. Thus, the objective function can be written as:

$$\min \alpha \sum_{s \in S} \sum_{i \in I} \sum_{l \in L} z_{il}^{s}(t) + \beta u(t) + \gamma k(t) + M \sum_{i \in I} (\widetilde{f}_i(t) + \widetilde{j}_i(t)) \tag{19}$$

In summary, the goal of the optimization problem is to minimize eqn. 19 subject to the constraints 3 - 18.

### C. Dynamic Traffic Engineering

Dynamic traffic engineering with a software-defined network is employed to manage traffic loads and enhance overall network performance. In the fog computing environment, traffic from the cluster points arrives randomly. In dynamic traffic engineering, resource allocation and release occur at each review point $t$, as shown in Fig.2. Review points $t \in T$ where $T$ represents a temporal window for dynamic traffic engineering. In the figure, the notation s_c_n and e_c_n denotes the start and end of the $n^{th}$ request from cluster point c. At each review point, the model is solved for incoming requests from the cluster points. With the help of the SDN controller our framework can be adopted to handle the real traffic as well. In case of real traffic, the optimization model will run at each review point inside the SDN controller as mentioned above. Then the resource will be allocated optimally to satisfy real traffic generated in fog computing.

## IV. Simulation Setup and Result Analysis

In our work, we formulated a MILP-based optimization problem and evaluated it using AMPL/CPLEX 12.10.0.0. Topology-related values used in our simulation are shown in Table III. According to [28], we have considered the fog layer-1 links' capacity as 100 Mbps, the fog layer-2 links' capacity as 70 Mbps, and 50 Mbps for the cloud. We considered dynamic traffic demand from different cluster points. Table IV shows values of different traffic demand from cluster points. Here, we considered a heterogeneous demand profile from four cluster points. For creating the maximum level of heterogeneity, we considered four demand profiles ⟨h1 , r1⟩ to ⟨h4 , r4⟩. The sum of bandwidth demand and server resource demand differs between sets by 12 Mbps and 3 GHz, respectively. We simulated the ⟨h , r⟩ tuple considering arrival rates from 0.75 to 2 with an increment of.25. The requests from different cluster points followed a Poisson distribution model. A lower arrival rate results in a larger time window, while a higher arrival rate indicates greater resource demand within a moderate time window. For our fog-cloud optimization model, selecting an arrival rate below 0.75 allows for observing significant findings over a longer time window. We began the simulation with an arrival rate of 0.75 and increased it to 2. Arrival rates above 2 led to a high blocking rate due to the simultaneous arrival of requests from different cluster points within a short time. We used five seeds for every arrival rate to justify our framework. Using shell scripting and the AWK programming language, we did our simulation and post-result analysis. We simulated our problem on a personal computer with a Linux operating system running on an Intel Core i5-8250U processor and 16 GB of memory.

We observed the change in bandwidth cost and average resource utilization with time for different arrival rates, while priority was given to bandwidth cost minimization, link-level load balancing, and server-level load balancing. We changed demands according to table IV and observed different variations in results. By accumulating all demand sets' results for different arrival rates, we formulated arrival rate-wise result variations. Due to the bursty nature of IoT traffic and shortage of resources at fog nodes, blocking happens in fog computing, so we have also figured out the blocking probability for different arrival rates and resource shortage types.

TABLE III
Topology Related Parameters

| # of CPs | 4 (1 in each region) | |
|---|---|---|
| Maximum # of available servers in each layer | Fog Layer-1 | 4 (each region) |
| | Fog Layer-2 | 8 |
| | Cloud | 40 |
| # of links/hops required to establish path from each CP | Fog Layer-2 | 10 |
| | Cloud | 20 |
| Per unit cost of bandwidth consumed from each link, $\varepsilon_l^s$ | $\varepsilon_l^{f1}$ | 1 |
| | $\varepsilon_l^{f2}$ | 2 |
| | $\varepsilon_l^{c}$ | 5 |
| Links' capacity (in Mbps) | Fog Layer-1 | 100 |
| | Fog Layer-2 | 70 |
| | Cloud | 50 |
| Capacity of each server (in GHz) | 2.5 | |

TABLE IV
Bandwidth and Resource Demand

| Bandwidth Demand (Mbps) | | | | |
|---|---|---|---|---|
| Notation | CP1 | CP2 | CP3 | CP4 |
| h1 | 1 | 3 | 5 | 7 |
| h2 | 10 | 8 | 6 | 4 |
| h3 | 7 | 9 | 11 | 13 |
| h4 | 16 | 14 | 12 | 10 |
| Resource Demand (GHz) | | | | |
| Notation | CP1 | CP2 | CP3 | CP4 |
| r1 | 0.25 | 0.75 | 1.25 | 1.75 |
| r2 | 2.5 | 2 | 1.5 | 1 |
| r3 | 1.75 | 2.25 | 2.75 | 3.25 |
| r4 | 4 | 3.5 | 3 | 2.5 |

### A. Changes in Bandwidth Cost and Resource Utilization for Different Arrival Rates

Fig. 3 shows the changes in bandwidth costs with time for different arrival rates, while priority is given to bandwidth cost minimization. From the figure, for higher arrival rates, the bandwidth costs are higher than for lower arrival rates. The figure is drawn except for the arrival rate of .75 which has a very long duration compared to other arrival rates.

In cases of lower arrival rates, requests persist for longer than at higher arrival rates. The arrival of requests per unit time is proportional to the arrival rates. At the high arrival rate, the request arrival time is short, so the generation of the same number of requests occurs in a short time as compared to the low arrival rate. In the event of lower arrival rates, bandwidth demands are mostly satisfied by the two fog layers, where the per-unit bandwidth cost for each link is low. As the two low-cost layers are heavily used, the bandwidth costs are low for

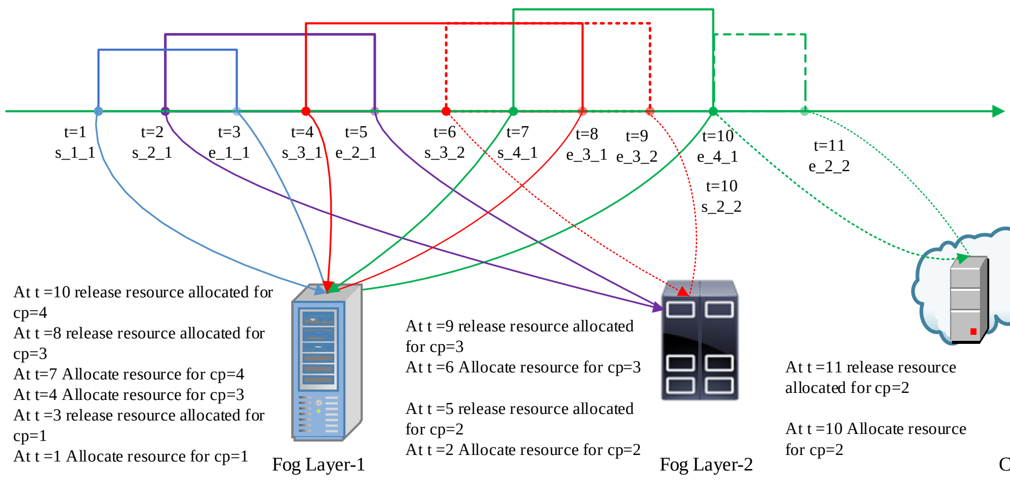
t=1
s_1_1
t=2
s_2_1
t=3
e_1_1
t=4
s_3_1
t=5
e_2_1
t=6
s_3_2
t=7
s_4_1
t=8
e_3_1
t=9
e_3_2
t=10
e_4_1
t=10
s_2_2
t=11
e_2_2
At t =10 release resource allocated for cp=4
At t =8 release resource allocated for cp=3
At t=7 Allocate resource for cp=4
At t=4 Allocate resource for cp=3
At t =3 release resource allocated for cp=1
At t =1 Allocate resource for cp=1
Fog Layer-1
At t =9 release resource allocated for cp=3
At t =6 Allocate resource for cp=3
At t =5 release resource allocated for cp=2
At t =2 Allocate resource for cp=2
Fog Layer-2
At t =11 release resource allocated for cp=2
At t =10 Allocate resource for cp=2
Cloud

Fig. 2. Dynamic traffic engineering framework in fog computing environment

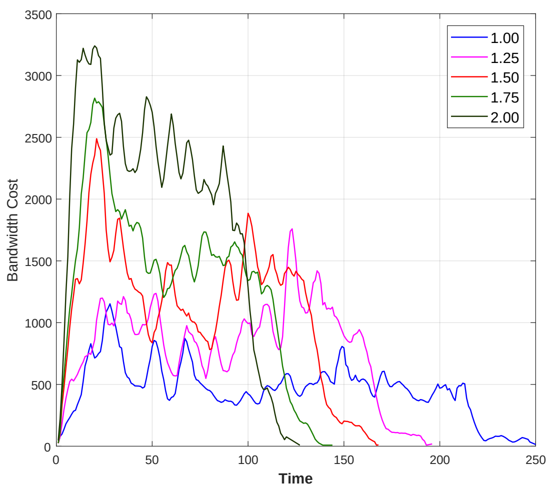

Fig. 3. Bandwidth cost with time for different arrival rates

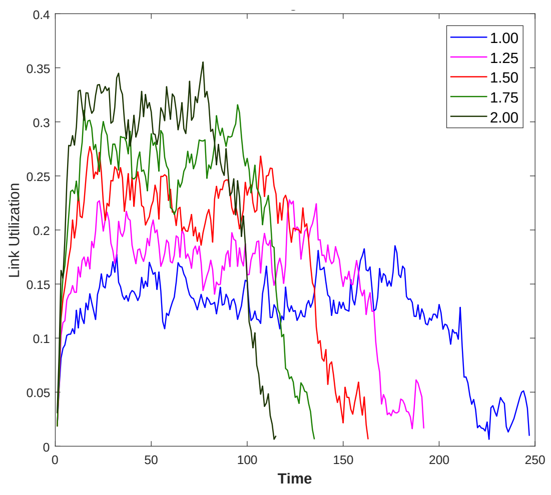

Fig. 4. Avg. link utilization with time for different arrival rates

lower arrival rates. From the different arrival rate curves, the lowest arrival rate of 1.00 has the lowest bandwidth cost of all the arrival rate curves.

For higher arrival rates, the resources of fog layer-1 and fog layer-2 are consumed very quickly, meaning the loads are transferred to the cloud within less time than for lower arrival rates. There is less scope for higher arrival rates to avoid the cloud layer. Due to the maximum usage of the costly cloud layer, bandwidth costs for higher arrival rates increase. Here, the maximum arrival rate of 2.00, which uses the cloud layer, mostly depicts the maximum bandwidth cost.

So, for designing a realistic fog computing architecture, the necessity of increasing the link bandwidth can be easily realised from the above discussions.

Fig.4 shows the changes in average link utilization over time for different arrival rates, with priority given to link-level load balancing. In this case, we emphasize the weight factor $\beta$. From the figure, it is clear that as the arrival rates increase, the values of average link utilization also increase.

For high arrival rates, all the requests arrive in a very short time compared to lower arrival rates. The involvement of links or usage of link resources is the highest. For the lowest arrival rate of 1.00 the maximum average link resource utilization is 18.50%. On the other hand, for the highest arrival rate, the value of maximum average link utilization is 35.20%.

Request arrival and mitigation make huge transition points on the curves. As the link capacity varies more than the per-link bandwidth cost for different layers, the transitions in the curves are more noticeable than the bandwidth cost curves.

Fig. 5 depicts the changes in average server utilization with time when priority is given to server-level load balancing. In this case, we emphasize the weight factor $\gamma$. The figure shows that as the arrival rates of requests increase, the servers' resource utilization also increases.

Server-level load balancing is performed to avoid over-

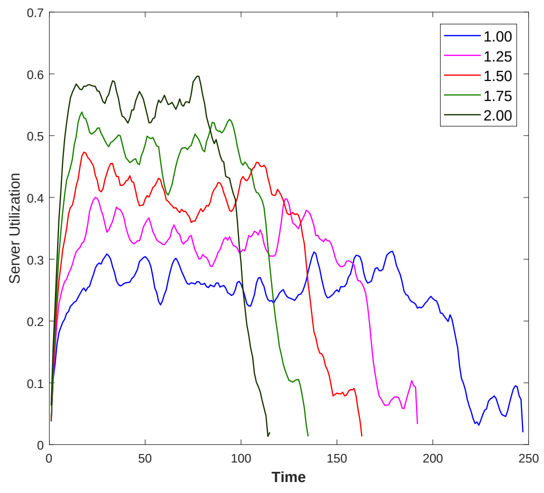

Fig. 5. Average server utilization with time for different arrival rates

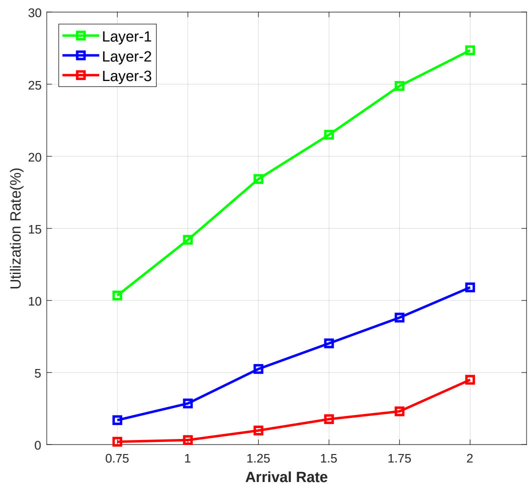

Fig. 6. Arrival rate versus link utilization for bandwidth cost minimization

utilization of any servers so that demands are distributed among possible destinations, resulting in fair resource utilization. For a lower arrival rate of $1.00$, the maximum average server utilization is $31.00\%$. On the other hand, for the highest arrival rate of $2.00$, the maximum average server utilization is $60.00\%$.

The average server utilization curves have fewer zigzag transition patterns than the average link utilization curves. The zigzag transition pattern depends on the differences in capacity of server resources among different layers. Server capacity is assumed to be the same across different layers and only varies the number of servers within each layer. As a result, the average server resource utilization curves are smoother than the time versus link utilization curves.

## *B. Layer-wise Resource Utilization with Time for Different Arrival Rates*

From the previous discussion, the fog architecture has three layers: fog layer-1, fog layer-2, and cloud/layer-3. Different layers' resources are used differently according to the optimization objectives.

Fig. 6 shows the usage of different layers' link resources for different arrival rates, while priority is given to bandwidth cost minimization. From the figure, fog layer-1 link utilization is higher than the other two layers. For bandwidth cost minimization, demands are mitigated within the lower fog layers rather than forwarding to the costly cloud layer.

From the figure, for lower arrival rates, the usage of the cloud layer is minimal and increases with arrival rates. Due to the nature of traffic at greater arrival rates, the use of the cloud layer is unavoidable. For the value of the highest arrival rate of $2.00$ Layer-1 link resource utilization is $27.00\%$, layer-2 utilization is $11.00\%$, but layer-3 utilization is only $4.00\%$.

Fig.7 depicts the usage of different layers' server resources while priority is given to bandwidth cost minimization. In this case, layer-1 servers' resources are used more than the other two layers for all arrival rates.

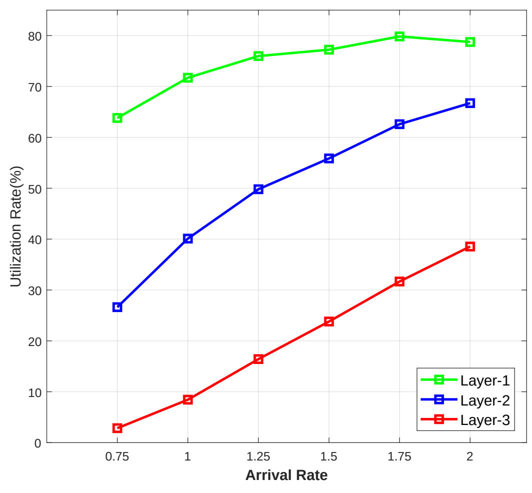

Fig. 7. arrival rate versus server resource utilization for bandwidth cost minimization

Bandwidth cost minimization aims to mitigate bandwidth demand from lower fog layers. Server resource utilization during bandwidth cost minimization maintains rationality in bandwidth allocation. According to the assumptions, network bandwidth resources are greater than server resources. From Fig. 6 and Fig. 7 for every arrival rate, it can be observed that servers' resource utilization percentages are higher than links' bandwidth utilization percentages.

Fig. 8 illustrates layer-wise link resource usage at various arrival rates with link-level load balancing. At lower arrival rates, link utilization is nearly equal across layers due to effective load distribution. At higher arrival rates, utilization varies because of the heterogeneity in link capacities and bursts of incoming requests.

Fig. 9 shows layer-wise server resource utilization for different arrival rates, while priority was given to link-level load balancing. The figure indicates proportionate resource

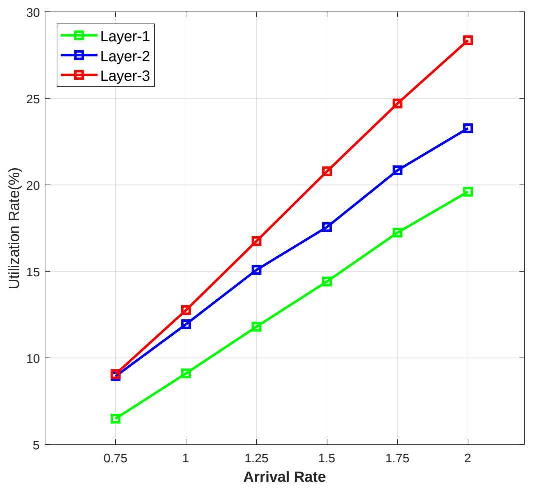

Fig. 8. Arrival rate versus layer-wise link utilization for link-level load balancing

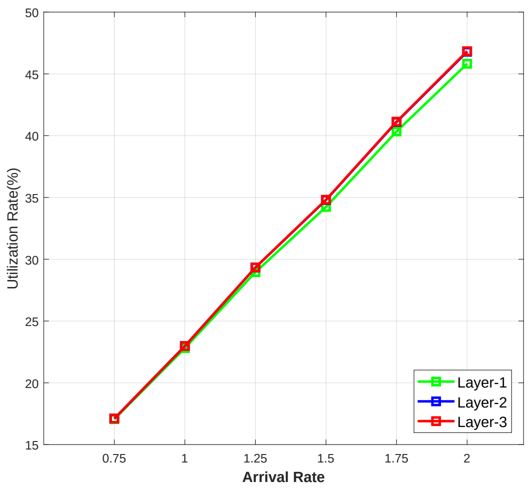

Fig. 10. Arrival rate versus layer-wise server resource utilization for server-level load balancing

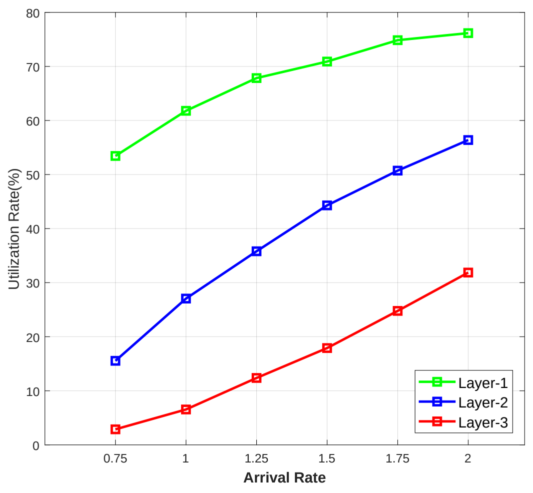

Fig. 9. Arrival rate versus layer-wise server resource utilization for link-level load balancing

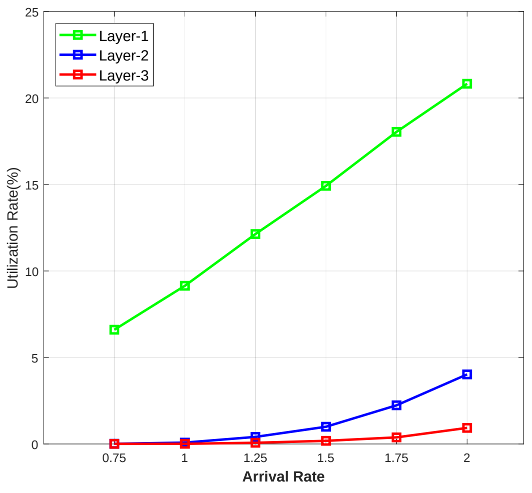

Fig. 11. Arrival rate versus layer-wise link utilization for server-level load balancing

allocation. During link-level load balancing, a proportionate resource allocation occurs. Layer-1 fog server resources are very low, so the proportionate bandwidth and server resource allocation show that layer-1 server resource utilization percentage is higher than the other two layers.

Fig. 10 depicts the changes of layer-wise server resource utilization for different arrival rates, while priority was given to server-level load balancing. From the figure, for lower and higher arrival rates, layer-wise server resource utilization is the same. Layer-1 and layer-2 server resource utilization curves coincide with each other. This happens because the purpose of server-level load balancing is to distribute loads among all servers to avoid overutilization of servers.

Fig. 11 depicts the changes of layer-wise link resource utilization for different arrival rates, while priority was given to server-level load balancing. The figure shows that the percentage of layer-1 link utilization is higher than the other two layers for every arrival rate. For server-level load balancing, priority is given to balancing loads among servers. So the three layers' server resource utilization is the same. Layer-wise link resource utilization is not equal due to the heterogeneity of links' capacity.

## C. *Optimization and blocking rates*

Blocking occurs when a request is rejected upon arrival due to insufficient bandwidth or server resources. In the formulated framework, blocking may happen in cases of resource shortages (network or server).

Fig. 12 shows the arrival rate versus blocking percentage, while priorities are given to bandwidth cost minimization, link-level load balancing, and server-level load balancing. From the figure, it can be observed that, as the arrival rate increases, the blocking percentage also increases.

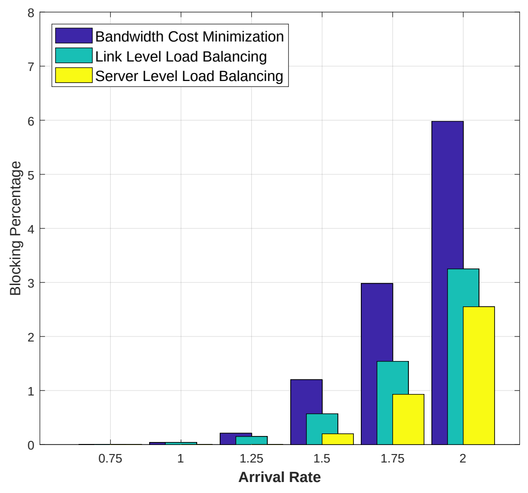

Fig. 12. Arrival rate versus blocking rate.

When priority is given to bandwidth cost minimization, the blocking percentage is higher than for the other two optimization objectives for every arrival rate. The second most blocking happens for link-level load balancing, and the blocking percentage is the minimum for server-level load balancing.

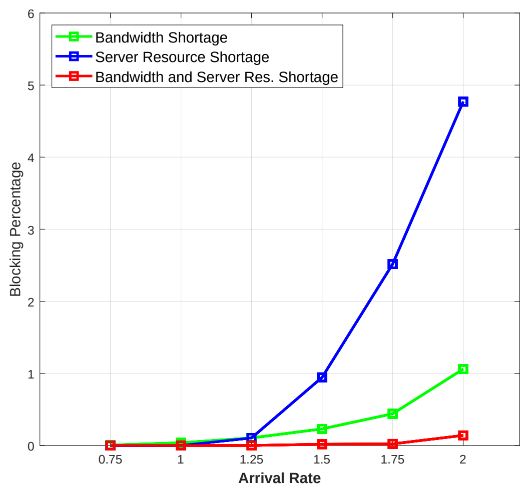

Fig. 13. Arrival rate versus blocking rate for bandwidth cost minimization.

Fig. 13 depicts the blocking percentage for various arrival rates when bandwidth cost minimization is prioritized. The figure also shows that for bandwidth cost minimization, blocking due to server resource shortage is higher than the link's bandwidth shortage.

Fig. 14 depicts the blocking percentage at different arrival rates while link-level load balancing is prioritized. Most of the blocking happens due to link bandwidth shortages, but blocking due to server resource shortages is also prominent. Link-level load balancing distributes loads among all possible links to avoid maximum or overutilization of any link. As a result, after a few times, new requests may experience blocking

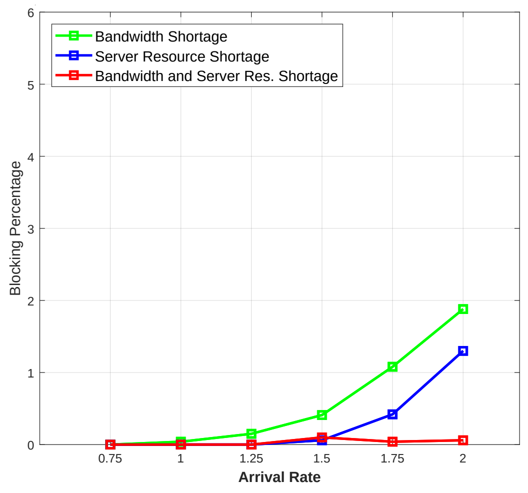

Fig. 14. Arrival rate versus blocking rate for link-level load balancing.

due to a link's bandwidth shortage. Blocking due to insufficient server resources is also common for higher arrival rates, as server-level load balancing is not prioritized during this link-level load balancing. The percentages are less than 2.00% for the highest arrival rate of 2.00.

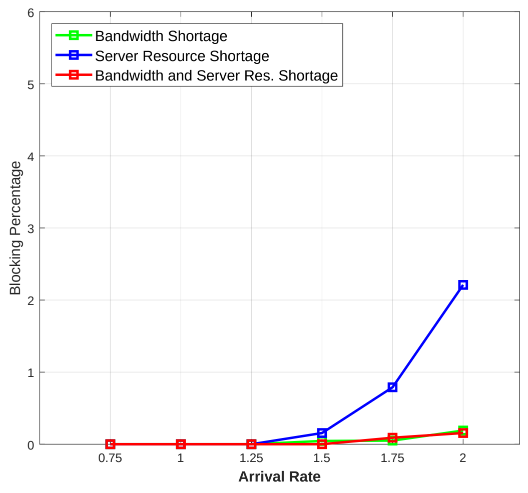

Fig. 15. Arrival rate versus blocking rate for server-level load balancing.

Fig. 15 shows blocking percentages for different arrival rates when priority was given to server-level load balancing. Here, most of the blocking happens due to servers' resource shortages rather than links' bandwidth shortages.Like link-level load balancing, a situation may occur where demands from any of the cluster points are discarded due to the non-availability of server resources.

### *D. Comparison Analysis: Prioritized Objectives vs. Baseline Framework*

Table V compares prioritized bandwidth cost minimization with the baseline framework, where no specific objective

TABLE V
COMPARISON OF FOG COMPUTING OPTIMIZATION WITH PRIORITIZED BANDWIDTH COST MINIMIZATION VS. BASELINE FRAMEWORK.

| Arrival Rate | Bandwidth Cost | Blocking Rate | Priority Type |
|---|---|---|---|
| 0.75 | 377.3 | 0 | Baseline Framework |
| 1 | 566.71 | 0.18 | |
| 1.25 | 984.36 | 0.32 | |
| 1.5 | 1084.01 | 1.61 | |
| 1.75 | 1427.67 | 2.74 | |
| 2 | 2169.62 | 5.27 | |
| 0.75 | 253.14 | 0 | Bandwidth Cost Minimization |
| 1 | 389.32 | 0.04 | |
| 1.25 | 704.44 | 0.21 | |
| 1.5 | 878.12 | 1.2 | |
| 1.75 | 1227.05 | 2.98 | |
| 2 | 1762.35 | 5.98 | |

is prioritized. The prioritized bandwidth cost minimization consistently yields lower average bandwidth costs across all arrival rates compared to the baseline framework. Overall, the prioritized approach results in a reduction of 24.07% in bandwidth costs. At higher arrival rates, we observe a slight increase in blocking rates. For bandwidth cost minimization, blocking due to server resource shortages increases at higher arrival rates.

Table VI illustrates the changes in link resource utilization between prioritized link-level resource utilization and the baseline framework. The baseline framework minimally utilizes the cloud layer, whereas it heavily relies on fog layer-1 in contrast to fog layer-2 and the cloud layer. Such unbalanced utilization typically leads to increased queuing delays in any network. The prioritized approach effectively distributes loads across various layers, contrasting with the baseline setup. Notably, at the lowest arrival rate of 0.75, the average fog layer link resource utilization has doubled, increasing from 3.64% to 7.71%. For the highest arrival rate of 2, the fog layer link utilization has increased by approximately 65%. The link-level load balancing has efficiently utilized both fog layer and cloud layer links. This balanced distribution among the available links across the three layers has significantly reduced the blocking rate.

A comparison between prioritized server-level load balancing and the baseline framework is presented in Table VII. The difference between fog layer and cloud layer resource utilization is notable in the baseline framework. As the arrival rate increases, cloud layer resource usage rises, leading to a reduction in the difference between fog layer and cloud layer server resource utilization. Similar to the link utilization shown in Table VI, the server resource utilization of fog layer-1 is higher than that of fog layer-2 and the cloud layer. In contrast, server-level load balancing maintains nearly uniform server resource utilization across different layers. The data from the table indicates that server-level load balancing has significantly reduced the blocking rate among different layers.

After reviewing Table V, Table VI, and Table VII, it becomes evident that server-level load balancing exhibits the lowest blocking rate. These findings underscore the importance of proper utilization of server resources in resource-constrained fog environments. Additionally, link-level load balancing plays a crucial role in reducing the blocking rate.

## V. CONCLUSION AND FUTURE WORK

A generalized three-layer fog computing system with dynamic traffic management techniques is proposed in this paper. Bandwidth cost minimization, link-level, and server-level load balancing are done here for the optimum usage of fog resources in a cooperative manner. Along with optimum utilization of fog resources, our proposed framework avoids overutilization of any link or server. The use of different layers for different optimization objectives is discussed with proper visualization. Blocking percentages for different arrival rates are shown in this paper, which is a significant factor for fog computing systems. To avoid blocking in a fog computing system, we also find out the remedy to combat blocking by finding the underlying factors. From our work, fog service providers can get a comprehensive idea about links' bandwidth and servers' resource allocation while designing a fog computing system to avoid unacceptable blocking in their system. We plan to execute our dynamic traffic model by adopting heuristics. Although our current work is simulation-based, the proposed model can be adopted in real traffic conditions. Our future work will involve testing with real IoT devices or traffic data. Additionally, we intend to introduce a machine learning algorithm for our proposed optimization model.

## REFERENCES

[1] Satish Narayana Srirama. A decade of research in fog computing: Relevance, challenges, and future directions. *Software: Practice and Experience*, 54(1):3–23, 2024.

[2] MR Poornima, HS Vimala, and J Shreyas. Holistic survey on energy aware routing techniques for iot applications. *Journal of Network and Computer Applications*, 213:103584, 2023.

[3] Mohammad Goudarzi, Huaming Wu, Marimuthu S Palaniswami, and Rajkumar Buyya. An application placement technique for concurrent iot applications in edge and fog computing environments. *IEEE Transactions on Mobile Computing*, 2020.

[4] Ayush Dongardive, Prasanna Fuse, Preeti Desai, Snehal Elkiwar, Gayatri Dere, and Ashutosh Marathe. Smart street light with power saving function and fault detection. In *2024 International Conference on Inventive Computation Technologies (ICICT)*, pages 1989–1994. IEEE, 2024.

[5] Cisco Systems, Inc. *Network Caching*. Cisco Systems, Inc, 2000.

[6] Mohammed Shuaib, Surbhi Bhatia, Shadab Alam, Raj Kumar Masih, Nayef Alqahtani, Shakila Basheer, and Mohammad Shabbir Alam. An optimized, dynamic, and efficient load-balancing framework for resource management in the internet of things (iot) environment. *Electronics*, 12(5):1104, 2023.

[7] Shatakshi Kokate and Urmila Shrawankar. Integration of the cloud with fog computing to secure data transmission between iot and cloud. In *Integration of Cloud Computing with Emerging Technologies*, pages 83–92. CRC Press, 2024.

[8] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, pages 37–42, 2015.

[9] Abhishek Hazra, Pradeep Rana, Mainak Adhikari, and Tarachand Amgoth. Fog computing for next-generation internet of things: fundamental, state-of-the-art and research challenges. *Computer Science Review*, 48:100549, 2023.

[10] Zoltán Ádám Mann. Notions of architecture in fog computing. *Computing*, 103(1):51–73, 2021.

[11] V Vijayakumar, Devarajan Malathi, Vairavasundaram Subramaniyaswamy, Palani Saravanan, and R Logesh. Fog computing-based intelligent healthcare system for the detection and prevention of mosquito-borne diseases. *Computers in Human Behavior*, 100:275–285, 2019.

TABLE VI
COMPARISON OF FOG COMPUTING OPTIMIZATION WITH PRIORITIZED LINK-LEVEL LOAD BALANCING VS. BASELINE FRAMEWORK.

| Arrival Rate | Avg. Fog Layer-1 Link Utilization | Avg. Fog Layer-2 Link Utilization | Avg. Fog Layer Link Utilization | Avg. Cloud Link Utilization | Blocking Rate | Priority Type |
|---|---|---|---|---|---|---|
| 0.75 | 7.26 | 0.02 | 3.64 | 0.01 | 0 | Baseline Framework |
| 1 | 10.05 | 0.11 | 5.08 | 0.06 | 0.18 | |
| 1.25 | 13.5 | 0.63 | 7.065 | 0.08 | 0.32 | |
| 1.5 | 16.24 | 1.66 | 8.95 | 0.28 | 1.61 | |
| 1.75 | 19.34 | 3.54 | 11.44 | 0.7 | 2.74 | |
| 2 | 22.23 | 5.27 | 13.75 | 2.01 | 5.27 | |
| 0.75 | 6.49 | 8.93 | 7.71 | 9.06 | 0 | link-level Load Balancing |
| 1 | 9.1 | 11.94 | 10.52 | 12.76 | 0.04 | |
| 1.25 | 11.8 | 15.09 | 13.45 | 16.75 | 0.15 | |
| 1.5 | 14.41 | 17.56 | 15.99 | 20.78 | 0.57 | |
| 1.75 | 17.24 | 20.84 | 19.04 | 24.7 | 1.54 | |
| 2 | 19.6 | 23.27 | 21.44 | 28.36 | 3.25 | |

TABLE VII
COMPARISON OF FOG COMPUTING OPTIMIZATION WITH PRIORITIZED SERVER-LEVEL LOAD BALANCING VS. BASELINE FRAMEWORK.

| Arrival Rate | Avg. Fog Layer-1 Server Res. Uti. | Avg. Fog Layer-2 Server Res. Uti. | Avg. Fog Server Res. Uti. | Avg. Cloud Server Res. Uti. | Blocking Rate | Priority Type |
|---|---|---|---|---|---|---|
| 0.75 | 39.04 | 33.37 | 36.21 | 7.46 | 0 | Baseline Framework |
| 1 | 44.53 | 38.70 | 41.61 | 13.7 | 0.18 | |
| 1.25 | 51.37 | 45.17 | 48.27 | 20.85 | 0.32 | |
| 1.5 | 54.55 | 48.45 | 51.5 | 26.99 | 1.61 | |
| 1.75 | 59.23 | 53.88 | 56.55 | 34.34 | 2.74 | |
| 2 | 63.61 | 58.76 | 61.19 | 41.45 | 5.27 | |
| 0.75 | 17.06 | 17.11 | 17.089 | 17.11 | 0 | server-level Load Balancing |
| 1 | 22.81 | 22.98 | 22.89 | 22.98 | 0 | |
| 1.25 | 28.96 | 29.33 | 29.15 | 29.34 | 0 | |
| 1.5 | 34.23 | 34.79 | 34.51 | 34.82 | 0.2 | |
| 1.75 | 40.34 | 41.1 | 40.72 | 41.14 | 0.93 | |
| 2 | 45.82 | 46.79 | 46.31 | 46.84 | 2.55 | |

[12] Azath Mubarakali, Anand Deva Durai, Mohmmed Alshehri, Osama AlFarraj, Jayabrabu Ramakrishnan, and Dinesh Mavaluru. Fog-based delay-sensitive data transmission algorithm for data forwarding and storage in cloud environment for multimedia applications. *Big Data*, 11(2):128–136, 2023.

[13] Guolong Chen, Liang Zhao, Xianwei Li, Fuqi Zhao, and Xiaojian Zeng. Optimal resource allocation for multimedia applications offloading in mobile edge computing. *IEEE Open Journal of the Computer Society*, 2:360–369, 2021.

[14] Hongyang Zhang and Rui Sun. A novel optimal management method for smart grids incorporating cloud-fog layer and honeybee mating optimization algorithm. *Solar Energy*, 262:111874, 2023.

[15] Hoan Le, Khaled Boussetta, and Nadjib Achir. A unified and semantic data model for fog computing. In *2020 Global Information Infrastructure and Networking Symposium (GIIS)*, pages 1–6. IEEE, 2020.

[16] Florian Metzger, Tobias Hoßfeld, André Bauer, Samuel Kounev, and Poul E Heegaard. Modeling of aggregated iot traffic and its application to an iot cloud. *Proceedings of the IEEE*, 107(4):679–694, 2019.

[17] Farah Ait Salaht, Frédéric Desprez, Adrien Lebre, Charles Prud'Homme, and Mohamed Abderrahim. Service placement in fog computing using constraint programming. In *2019 IEEE International Conference on Services Computing (SCC)*, pages 19–27. IEEE, 2019.

[18] Hatem A Alharbi, Taisir EH El-Gorashi, and Jaafar MH Elmirghani. Energy efficient virtual machine services placement in cloud-fog architecture. In *2019 21st International Conference on Transparent Optical Networks (ICTON)*, pages 1–6. IEEE, 2019.

[19] Shohei Kamamura. Dynamic traffic engineering considering service grade in integrated service network. *IEEE Access*, 10:79021–79028, 2022.

[20] Nitin Varyani, Zhi-Li Zhang, and David Dai. Qroute: An efficient quality of service (qos) routing scheme for software-defined overlay networks. *IEEE Access*, 8:104109–104126, 2020.

[21] Shalli Rani, Divya Gupta, Norbert Herencsar, and Gautam Srivastava. Blockchain-enabled cooperative computing strategy for resource sharing in fog networks. *Internet of Things*, 21:100672, 2023.

[22] Hoa Tran-Dang and Dong-Seong Kim. Cooperation for distributed task offloading in fog computing networks. *Cooperative and Distributed Intelligent Computation in Fog Computing: Concepts, Architectures, and Frameworks*, pages 33–45, 2023.

[23] Xinchen Lyu, Chenshan Ren, Wei Ni, Hui Tian, and Ren Ping Liu. Cooperative computing anytime, anywhere: Ubiquitous fog services. *IEEE Wireless Communications*, 27(1):162–169, 2020.

[24] Nour Mostafa. Cooperative fog communications using a multi-level load balancing. In *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 45–51. IEEE, 2019.

[25] Yukai Hou, Zhiwei Wei, Rongqing Zhang, Xiang Cheng, and Liuqing Yang. Hierarchical task offloading for vehicular fog computing based on multi-agent deep reinforcement learning. *IEEE Transactions on Wireless Communications*, 2023.

[26] Mostafa Haghi Kashani and Ebrahim Mahdipour. Load balancing algorithms in fog computing. *IEEE Transactions on Services Computing*, 16(2):1505–1521, 2022.

[27] Alireza Sadeghi Milani and Nima Jafari Navimipour. Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *Journal of Network and Computer Applications*, 71:86–98, 2016.

[28] Guangshun Li, Yonghui Yao, Junhua Wu, Xiaoxiao Liu, Xiaofei Sheng, and Qingyan Lin. A new load balancing strategy by task allocation in edge computing based on intermediary nodes. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):1–10, 2020.

[29] Song Ningning, Gong Chao, An Xingshuo, and Zhan Qiang. Fog computing dynamic load balancing mechanism based on graph repartitioning. *China Communications*, 13(3):156–164, 2016.

[30] Simar Preet Singh, Rajesh Kumar, Anju Sharma, Jemal H Abawajy, and

Ravneet Kaur. Energy efficient load balancing hybrid priority assigned laxity algorithm in fog computing. *Cluster Computing*, 25(5):3325–3342, 2022.

[31] Abrar Saad Kadhim and Mehdi Ebady Manaa. Hybrid load-balancing algorithm for distributed fog computing in internet of things environment. *Bulletin of Electrical Engineering and Informatics*, 11(6):3462–3470, 2022.

[32] Muzammil Hussain Shahid, Ahmad Raza Hameed, Saif ul Islam, Hasan Ali Khattak, Ikram Ud Din, and Joel JPC Rodrigues. Energy and delay efficient fog computing using caching mechanism. *Computer Communications*, 2020.

[33] Bushra Jamil, Mohammad Shojafar, Israr Ahmed, Atta Ullah, Kashif Munir, and Humaira Ijaz. A job scheduling algorithm for delay and performance optimization in fog computing. *Concurrency and Computation: Practice and Experience*, 32(7):e5581, 2020.

[34] Mohamed Abd Elaziz, Laith Abualigah, and Ibrahim Attiya. Advanced optimization technique for scheduling iot tasks in cloud-fog computing environments. *Future Generation Computer Systems*, 124:142–154, 2021.

[35] Asad Waqar Malik, Anis Ur Rahman, Tariq Qayyum, and Sri Devi Ravana. Leveraging fog computing for sustainable smart farming using distributed simulation. *IEEE Internet of Things Journal*, 7(4):3300–3309, 2020.

[36] Mirza Mohd Shahriar Maswood, MD Rahinur Rahman, Abdullah G Alharbi, and Deep Medhi. A novel strategy to achieve bandwidth cost reduction and load balancing in a cooperative three-layer fog-cloud computing environment. *IEEE Access*, 8:113737–113750, 2020.

[37] William Tichaona Vambe. Fog computing quality of experience: Review and open challenges. *International Journal of Fog Computing (IJFC)*, 6(1):1–16, 2023.

[38] MLM Peixoto, E Mota, AHO Maia, W Lobato, MA Salahuddin, R Boutaba, and LA Villas. Fogjam: A fog service for detecting traffic congestion in a continuous data stream vanet. *Ad Hoc Networks*, 140:103046, 2023.

[39] Alzahraa Elsayed, Khalil Mohamed, and Hany Harb. Enhanced traffic congestion management with fog computing: A simulation-based investigation using ifog-simulator. *arXiv preprint arXiv:2311.01181*, 2023.

[40] Mirza Mohd Shahriar Maswood, Robayet Nasim, Andreas J Kassler, and Deep Medhi. Cost-efficient resource scheduling under qos constraints for geo-distributed data centers. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2018.

[41] Ruiyun Liu, Weiqiang Sun, and Weisheng Hu. Placement of high availability geo-distributed data centers in emerging economies. *IEEE Transactions on Cloud Computing*, 2023.

[42] Paulo Maciel, Jamilson Dantas, Carlos Melo, Paulo Pereira, Felipe Oliveira, Jean Araujo, and Rubens Matos. A survey on reliability and availability modeling of edge, fog, and cloud computing. *Journal of Reliable Intelligent Environments*, pages 1–19, 2021.

[43] Nader Daneshfar, Nikolaos Pappas, Valentin Polishchuk, and Vangelis Angelakis. Service allocation in a mobile fog infrastructure under availability and qos constraints. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[44] Mirsaeid Hosseini Shirvani and Yaser Ramzanpoor. Multi-objective qos-aware optimization for deployment of iot applications on cloud and fog computing infrastructure. *Neural Computing and Applications*, 35(26):19581–19626, 2023.

[45] G Shruthi, Monica R Mundada, S Supreeth, and Bryan Gardiner. Deep learning-based resource prediction and mutated leader algorithm enabled load balancing in fog computing. *International Journal of computer networks and information security*, 15(4):84–95, 2023.

[46] Linh-An Phan, Duc-Thang Nguyen, Meonghun Lee, Dae-Heon Park, and Taehong Kim. Dynamic fog-to-fog offloading in sdn-based fog computing systems. *Future Generation Computer Systems*, 117:486–497, 2021.

[47] Sujit Bebortta, Subhranshu Sekhar Tripathy, Umar Muhammad Modibbo, and Irfan Ali. An optimal fog-cloud offloading framework for big data optimization in heterogeneous iot networks. *Decision Analytics Journal*, page 100295, 2023.

[48] Keigo Mukae, Takumi Saito, Shigenari Nakamura, Tomoya Enokido, and Makoto Takizawa. Design and implementing of the dynamic tree-based fog computing (dtbfc) model to realize the energy-efficient iot. In *Advances in Internet, Data and Web Technologies: The 9th International Conference on Emerging Internet, Data & Web Technologies (EIDWT-2021)*, pages 71–81. Springer, 2021.

[49] Faten A Saif, Rohaya Latip, Zurina Mohd Hanapi, and Kamarudin Shafinah. Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing. *IEEE Access*, 11:20635–20646, 2023.

[50] Saurabh Singhal, Senthil Athithan, Madani Abdu Alomar, Rakesh Kumar, Bhisham Sharma, Gautam Srivastava, and Jerry Chun-Wei Lin. Energy aware load balancing framework for smart grid using cloud and fog computing. *Sensors*, 23(7):3488, 2023.

[51] Muhammad Babar Kamal, Nadeem Javaid, Syed Aon Ali Naqvi, Hanan Butt, Talha Saif, and Muhammad Daud Kamal. Heuristic min-conflicts optimizing technique for load balancing on fog computing. In *Advances in Intelligent Networking and Collaborative Systems: The 10th International Conference on Intelligent Networking and Collaborative Systems (INCoS-2018)*, pages 207–219. Springer, 2019.

[52] Neco Villegas, Luis Diez, Idoia De la Iglesia, Marco González-Hierro, and Ramón Agüero. Energy-aware optimum offloading strategies in fog-cloud architectures: A lyapunov based scheme. *IEEE Access*, 2023.

[53] Eva Marín Tordera, Xavi Masip-Bruin, Jordi Garcia-Alminana, Admela Jukan, Guang-Jie Ren, Jiafeng Zhu, and Josep Farré. What is a fog node a tutorial on current concepts towards a common definition. *arXiv preprint arXiv:1611.09193*, 2016.

**Md. Rahinur Rahman** received the B.Sc. degree in electronics and communication engineering from the Khulna University of Engineering & Technology, Bangladesh, in 2016. He also completed the M.Sc. degree in electronics and communication engineering from the Khulna University of Engineering & Technology, Bangladesh, in 2022. He is also working as an Assistant Director (ICT) with the Department of Information and Communication Technology, Bangladesh Bank (The Central Bank of Bangladesh). His research interests include future networking, the IoT, fog computing, and cyber security.

**Mirza Mohd Shahriar Maswood (Member, IEEE)** University of Engineering & Technology (KUET), Bangladesh, in 2010, and the M.Sc. degree in electrical engineering and the Ph.D. degree in telecommunications and computer networking from the University of Missouri Kansas City (UMKC), USA, in 2015 and 2018, respectively. During his Ph.D. at UMKC, he worked as a Graduate Research Assistant. He is currently an Associate Professor with the Department of Electronics and Communication Engineering, KUET. He has served as a reviewer for the Journal of Network and Systems Management and IEEE Transactions on Parallel and Distributed Systems. He reviewed more than 40 peer-reviewed conference and journal articles. His research interests include data center optimization, traffic engineering, cloud computing, fog computing, and machine learning.