information

Article

BDPS: An Efficient Spark-Based Big Data Processing Scheme for Cloud Fog-IoT Orchestration

Rakib Hossen [1], Md Whaiduzzaman [2,3,*], Mohammed Nasir Uddin [1], Md. Jahidul Islam [1], Nuruzzaman Faruqui [2], Alistair Barros [3], Mehdi Sookhak [4] and Md. Julkar Nayeen Mahi [2]

[1] Department of Computer Science and Engineering, Jagannath University, Dhaka 1100, Bangladesh; rakibcsejnu001@gmail.com (R.H.); nasir@cse.jnu.ac.bd (M.N.U.); jahid@cse.green.edu.bd (M.J.I.)
[2] Institute of Information Technology, Jahangirnagar University, Dhaka 1342, Bangladesh; faruquizaman27@gmail.com (N.F.); mahi.1992@gmail.com (M.J.N.M.)
[3] School of Information Systems, Queensland University of Technology, Brisbane 4000, Australia; alistair.barros@qut.edu.au
[4] Department of Computer Science, Texas AM University, Corpus Christ, TX 78412, USA; m.sookhak@ieee.org
* Correspondence: wzaman@juniv.edu

Abstract: The Internet of Things (IoT) has seen a surge in mobile devices with the market and technical expansion. IoT networks provide end-to-end connectivity while keeping minimal latency. To reduce delays, efficient data delivery schemes are required for dispersed fog-IoT network orchestrations. We use a Spark-based big data processing scheme (BDPS) to accelerate the distributed database (RDD) delay efficient technique in the fogs for a decentralized heterogeneous network architecture to reinforce suitable data allocations via IoTs. We propose BDPS based on Spark-RDD in fog-IoT overlay architecture to address the performance issues across the network orchestration. We evaluate data processing delays from fog-IoT integrated parts using a depth-first-search-based shortest path node finding configuration, which outperforms the existing shortest path algorithms in terms of algorithmic (i.e., depth-first search) efficiency, including the Bellman–Ford (BF) algorithm, Floyd–Warshall (FW) algorithm, Dijkstra algorithm (DA), and Apache Hadoop (AH) algorithm. The BDPS exhibits low latency in packet deliveries as well as low network overhead uplink activity through a map-reduced resilient data distribution mechanism, better than in BF, DA, FW, and AH. The overall BDPS scheme supports efficient data delivery across the fog-IoT orchestration, outperforming faster node execution while proving effective results, compared to DA, BF, FW and AH, respectively.

Keywords: efficient data processing; depth-first search; map reduction; in-memory accelerator; spark

Citation: Hossen, R.; Whaiduzzaman, M.; Uddin, M.N.; Islam, M.J.; Faruqui, N.; Barros, A.; Sookhak, M.; Mahi, M.J.N. BDPS: An Efficient Spark-Based Big Data Processing Scheme for Cloud Fog-IoT Orchestration. *Information* **2021**, *12*, 517. https://doi.org/10.3390/info12120517

Academic Editor: Willy Susilo

Received: 4 November 2021
Accepted: 7 December 2021
Published: 10 December 2021

1. Introduction

The growing IoT network [1–3], end-to-end packet distribution, finds mesh networks for end devices, along with short-range communication [4–6]. The anonymous users and excessive delays in heterogeneous network hops make it challenging to deliver data with better scalability and elasticity [7–9]. Increasing user loads within IoT connections raises latency-based delays toward gateway endpoints [10–12]. When distributed processes have been merged and aligned to continue [13,14] through the network gateway, they occur due to delayed package delivery and inconsistent planning of processes for high memory gain at the edge of the network nodes [15,16].

Fog computing reduces latency problems by distributing processes across different shared hops, splitting them at their right ends, and assigning network data within a timed frame (i.e., time-based or peer-to-peer programming manner). The end devices or the IoT network receive their collected data by managing a lower latency period and sharing defined (i.e., planned) workloads in federated fog-IoT orchestrations [17]. In connection with these IoT gateways, custom fogs achieve a lower memory gain through less delay-based data provision [18,19] across these IoT gateways. When processing large amounts of data (e.g., big data), IoT gateways experience problems with the distribution to the intended users, and errors occur during data processing. The middleware fog architecture of the middleware can also help reduce the overhead of cloud data processing as the data delivery (i.e., mapping) is loaded onto validated users [20].

Typical big data architecture works with Apache Hadoop-based data mapping deliveries concerning veracity, validity, velocity, variability, volatility, and vulnerability. However, Hadoop makes it difficult and increases additional latency-based delay when considering validity, variability, and volatility across data delivery mappings [21]. Hadoop is highly volatile, and hence, a resilient data delivery mechanism is a reasonable need for the appropriate data validation. Thus, for efficient data delivery at the IoT hop ends, a resilient data distribution-based mapping mechanism [22] is a better choice, and fog service deliverables (i.e., gateways, and microservice extenders) act here as a beneficiary helper for the three-tier architecture [23].

The overlay tree topology for interconnects works as mesh-based peer-to-peer data sharing [24,25], a master–slave and slave–master pattern. IoT networks often encounter a delay in the two-way verified communication of latencies [26] at the hop-to-hop ends offloading [27], causing latencies based on delay at the network ends [28] of the devices depending on their data delivery priorities, end-to-end [29,30]. Fog networks have difficulty in making appropriate decisions for end-to-end data distribution. Therefore, process planners at the fog endpoints [31,32] must access and share through the IoT data allocation procedures. A valid developed schema based on proper planning resource allocation can fine-tune fog services [33], enabling integrated fog-IoT orchestrations. Without the verification from the cloud, the fog cluster cannot validate the scheduled data distribution. Hence, for rapid data delivery and efficient in-memory scheduling across the network ends, a low latency-based data distribution scheme is still needed for fog-IoT overlay networks.

We propose BDPS, a delay optimization scheme to run with in-memory Spark processing to reduce the Apache Hadoop [34] extra delay problem for the verification, validation, and execution of files caused by distribution of data processing. "BDPS" can manage data processing [35] for architectures or massive storage modules to fast process scheduling to run different frame intervals in a single frame slot and minimize the end-to-end hop latency problem. Deploying the Spark framework activity through resilient distributed data can help to speed up deliveries of data within the overlay network and improve network efficiency by reducing the extended delay problem on mesh interconnections at the hop-to-hop ends. A star mesh topology was designed to connect the main fog controller via cloud for faster scaling of the network delay measurements and process validations [36]. The cloud is at the back end of our developed BDPS scheme, and gateways of IoTs are placed at the front end [37]. In the fog layer [38], the RAMA scheduling process runs through BDPS data distribution [39], containers and delivering priority scheduled data [40] from the IoT layer toward the central cloud [41]. After the fog cluster [42,43] validates the frames [44–46], the derived data feed to the RDD in memory accelerators to provide efficient data deliveries across the IoT clusters [47,48] (i.e., gateways).

This paper uses three popular shortest-path-routing algorithms (i.e., the Bellman–Ford (BF) algorithm, Floyd–Warshall (FW) algorithm and Dijkstra algorithm (DA)) and a map-reduction data delivery algorithm (i.e., Apache Hadoop (AH) algorithm) based comparison (Table 1) to estimate a performance analysis. The performance analysis is shown by developing a rapid data delivery scheme (i.e., BDPS) across the fog-IoT environment. However, the packet drop ratio, frame delivery ratio, network overhead, computational delay, are throughput are primarily checked to estimate the performance analysis within a hierarchical tree architecture.

Table 1. List of acronyms used in alphabetical order.

Notations	Description
APSP	ALL pair shortest path
AH	Apache Hadoop algorithm
BF	Bellman–Ford algorithm
DA	Dijkstra algorithm
DFS	Depth-first search algorithm
FW	Floyd–Warshall algorithm
IoT	Internet of Things
BDPS	Spark-RDD-in-memory-based scheme
RAMA	Read-only memory accelerator
RDD	Resilient data distributor
SP	Session password

The main contributions of the research work include the following:

- We present an efficient Spark-RDD-in-memory-based scheme to mitigate against existing challenges in a scheduled priority batch processing data distribution framework.
- We implement the delay efficient high-speed data processing algorithmic scheme for a hierarchical tree-based overlay mesh architecture in the cloud–fog and IoT ecosystem.
- We conduct an overall performance analysis of our scheme with shortest path algorithms, such as BF, FW and AH, with efficient delay time analysis (e.g., system run-time tolerance or service load).

The remainder of this paper is organized as follows: Section 2 generalizes a discussion about BDPS scheme work procedures within the fog-IoT environment. Section 3 employs an overview of the background study to identify the current solutions and limitations. Section 4 represents the system model and methodology of the developed BDPS scheme. Section 5 illustrates a rigorous experimental result of BDPS and other existing algorithms. Section 6 concludes the paper by providing future insights and outcomes.

2. Background Study with Related Work

In this section, we discuss the background study of fog-IoT, SDN and Spark-RDD related to this research.

2.1. Spark-RDD

Spark has RDDs (i.e., resilient distributed datasets), performed as the main unit of logical data structures. RDDs are mainly distributed objects, and depend on storing data toward memory locations (i.e., cache and RAM) or on disks attached over machines within a cluster. A single RDD may be divided into multiple partitions of logical data slots. These configured partitions can store and process over different clusters of the allocated machines. RDD data sets are read-only (i.e., immutable) in nature but process override may create new RDD stacks within previous stacks through granular operations.

The RDDs can be stored in the cache and reusable. RDDs in Spark (Figure 1) are estimated in a lazy order (i.e., evaluates the processing delay until it is needed to be overlapped since it carries memory-based operations). This procedure saves a good amount of process execution time and shows effectiveness across the network. In Spark, the RDDs may operate in two ways—(1) transformations and (2) actions.

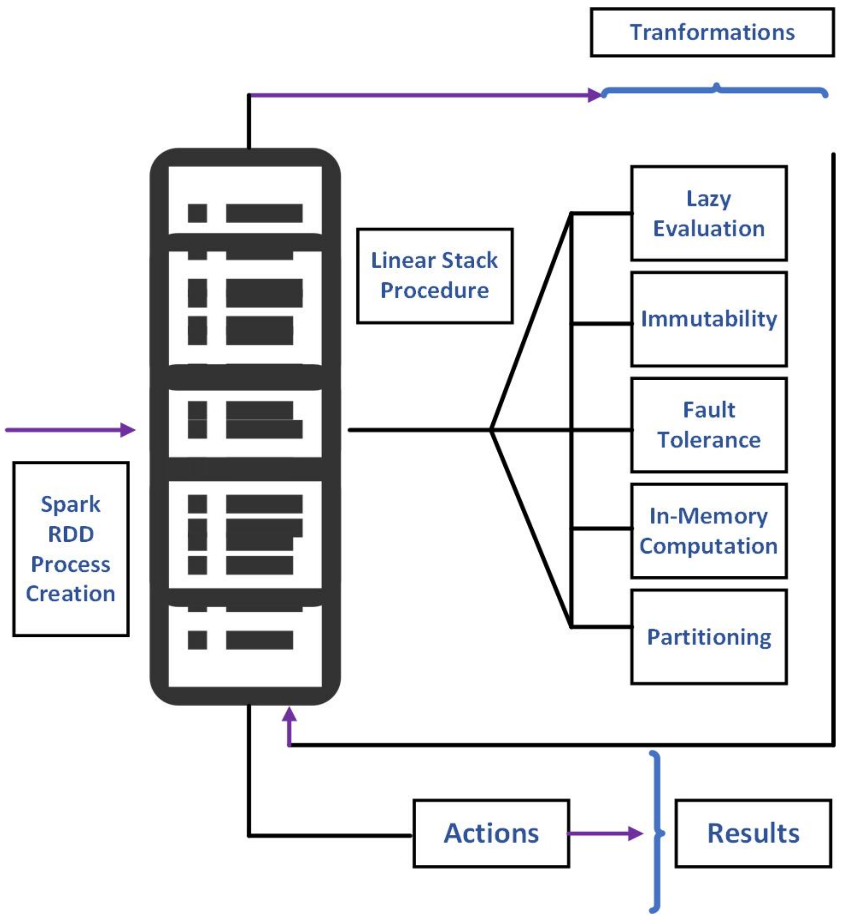

Figure 1. Spark-RDD mechanism.

2.2. SDN

Software-defined network (i.e., SDN) architecture defines a computing system of networks building through a distinct set of open-software technologies and hardware commodities. The SDN architecture works by dividing the data and control plane of the networked stacks packaged and integrated with code distribution generated by different vendor proprietaries. The SDN uses an OpenFlow protocol for the whole network, and preferably considers the controller plane. The SDN data controller gets instructions from the application layer and employs them toward active components through application programming interfaces (APIs). The SDN APIs are defined as north- and south-bound interfaces carrying virtual network overlays, and the allocated devices need not be physically located at the same place while performing operations.

2.3. Fog-IoT

Fog computing is a decentralized communication infrastructure, where computational data, applications, and storage reside between the source data and the cloud counterparts. Fog communication supersedes cloud computation and permits short data analytics near the edge, whereas the cloud employs long-term resource-intensive analytics. Internet-of-things (i.e., IoT) fog computation reveals the initial benefits and power enhancements of the cloud nearer to a middle layer of communication (Figure 2) in which the data are created and acted as per the strategy build-ups. Moreover, IoT devices and actuators are the platforms where data are gathered and updated, but they do not have high computational and storage capabilities to engage artificial intelligent tasks or advance analytics.

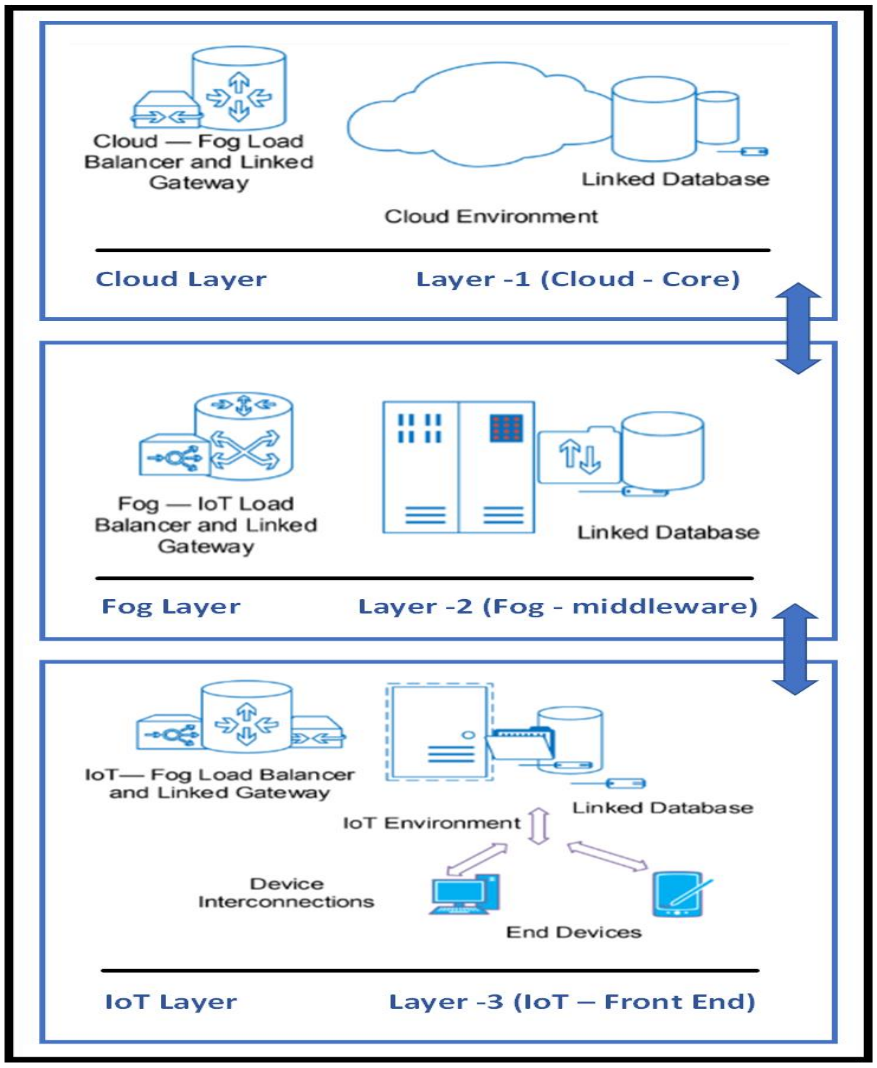
Cloud — Fog Load Balancer and Linked Gateway
Linked Database
Cloud Environment
Cloud Layer
Layer -1 (Cloud - Core)
Fog — IoT Load Balancer and Linked Gateway
Linked Database
Fog Layer
Layer -2 (Fog - middleware)
IoT— Fog Load Balancer and Linked Gateway
Linked Database
IoT Environment
Device Interconnections
End Devices
IoT Layer
Layer -3 (IoT — Front End)

Figure 2. Fog-IoT orchestration.

The approaches of fogging address the emerging issues of IoT and industrial IoT (i.e., IIoT) by reducing computation delays, communication latencies between sensors and the cloud, presumably affecting IoT performance, and providing effectiveness across the network.

2.4. Overlay Tree Architecture

The overlay tree architecture deploys an internal tree of communication processes to leverage the application of processes toward the end-users. In the root of the tree architecture, there resides the front-end process, and at the leaves, there are the back-end processes to be carried by end-points within tree architecture. The processes run through as a first-in-first-out or last-in-first-out manner based on priority queues within the tree nodes. The data abstraction filter is generated for data aggregation and reduction operations over in-stack packet deliveries. The persistent filter carries side effects of filter execution from one to the next and increases filter abstraction availability. The overlay tree architecture communicates with a master–slave and slave–master pattern (Figure 3) with its respective parent–child nodes within a hierarchical fashion. This communication approach provides an efficient reliability mechanism across the parent–child data delivery process.

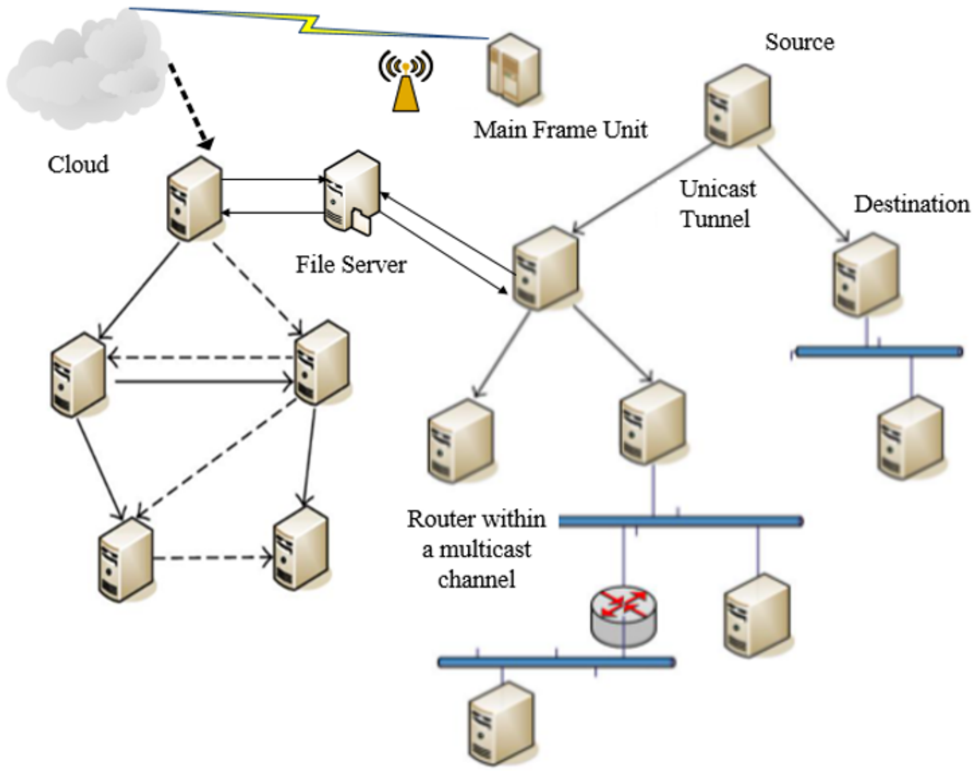

Figure 3. A generic mesh-based overlay tree architecture.

3. Related Works

Proper data acquisition and distribution is a vital concern for heterogeneous fog-IoT networks. It is necessary to orchestrate and tune the interaction of processes in a resilient distributed data mechanism to provide efficient data deliveries over the fog–cloud data flow while ensuring high-speed data processing.

Ramya et al. [4] found out the lightweight dynamic graph query mechanisms for the networking-as-a-service (i.e., NaaS) system, which provide access to cloud customers to virtual network functions. They used the net graph library to support network management operations and designed an architecture using the all pair shortest path (i.e., APSP) algorithm to pre-compute all topology paths and achieve practical compute times.

Whaiduzzaman, M., Naveed, A and Gani, A. [5] implemented a cloudlet resource enhancement framework, MobiCore. MobiCore uses the M/M/c/K queue, and the system model runs through the birth death Markov chain execution of processes. The framework supports an optimal average service time 1/μ of cloudlets for the mobile applications to outperform the maximum benefit. The MobiCoRE framework gathers up to 50% extra users while performing optimal service time and shared mobile resources for task remaining within the cloudlet. It can also achieve up to 47% time enhancement for mobile device users in terms of sharing only 16% resources of the cloudlet.

EL-Garoui, L., Pierre, S. and Chamberland, S. [7] proposed a new routing protocol based on SDN and the Naive Bayes framework to improve the delay and also minimize the duration between the sender and receiver. They achieved realistic results using a machine learning algorithm and designed a new routing protocol to formulate the delay.

Ragaventhiran, J. and Kavithadevi, M.K. [11] constructed a map reduce optimization framework through the emperor penguin colony (EPC) approach while comparing the mapper and reducer. The optimization framework is an extended version of frequent pattern mining (FPM), primarily allocated in the Hadoop cluster to reduce data redundancy. The framework works in canonical order (CAN) tree-based frequent pattern (FP) growth to reduce the data execution time and frequent tree construction while performing within the Hadoop cluster. The affinity propagation (AP) based clustering of this optimization framework helps to avoid oversight of data scanning and growing minimal searching space in MapReduce to deploy balanced loads among different data blocks. The performance evaluation of this framework with the existing works supports the succeeding metrics for execution time, scalability, response time and load balancing rate.

Whaiduzzaman et al. [12] developed a fault-tolerant framework for seamless microservice execution within the fog-IoT orchestration. The framework supports efficient task execution of end-to-end processes while performing the seamless transfer of microservices toward the intended fog-IoT shared devices.

Awan et al. [13] developed a procedure for detecting and relieving DDoS attacks through using Apache Spark from a large amount of data delivery times. In their research outcomes, the Apache Spark process shows much faster data execution than other existing frameworks. They used SDN (i.e., software-defined networking) to visualize their process of work. In our work, we developed an in-memory processing-based Spark-RDD high-speed data delivery scheme for efficient computation and providing less delay across the network hops or ends. Wilfried et al. [15] proposed a MapReduce framework that promotes computing the shortest path routing mechanism into a large-scale graph of a parallel and distributed algorithm referred to as A* algorithm. The algorithm comprises a large road network into some small path network. However, this proposed algorithm reduces the time complexity while showing not many effective results.

Quasim, M.T. [16] employed a mobile edge computing-based resource management and task scheduling (MEC-RMTS) framework to deliver an efficient task-offloading-based data delivery mechanism with low latency and scalability for IoT-network-enabled smartphone handling. The developed module uses power usage (PU) to decrease energy consumption, providing a convex technique of gradient-dependent game model (GM) data acquisition. Moreover, a mixed-intelligent, non-linear program model is configured for the issue of common needs and resource utilization (CNRU) to minimize request delays at the end device nodes. The simulated performance depicts that MEC-RMTSF saves the energy usage by up to 7.1% and gathers a delay of 0.95 seconds for scheduled resource allocation.

Dongbo et al. [17] proposed an algorithmic mechanism to suppress a large-scale road network by dividing into some small-scale road networks according to the configured central distance. This algorithmic procedure creates better executing results.

Aslanpour, M.S., Gill, S.S. and Toosi, A.N. [19] illustrated an in-depth performance analysis review considering the latest technology changes within cloud, fog and IoT process optimizations, device interactions, etc. The descriptive review shows possible future technology development taxonomies, benchmarks and standardization.

Abdel-Basset et al. [24] used a marine predators algorithm (MPA) to improve the quality of service of task scheduling across the IoT networks. However, fog distributions are triggered to carry genetic mutations of process reinitialization, specifically to compute and deliver the best packet stacks throughout the fog-IoT orchestrated environment. The MPA is compared with other existing metaheuristic algorithms and genetic algorithms to show effective performance metrics, considering energy consumption, makespan, flow time, and the carbon dioxide emission rate within the fog-IoT device allocations.

Zhihui et al. [29] described a big data edge cloud data analytics model to develop an IoTDeM big data feasibility model which suppresses the existing LWLR model in multiple edge clouds of the Hadoop 1 and Hadoop 2 environment. The generalized IoTDeM model selects cluster scale parameters to improvise the traditional LWLR model and finds effective results in data reliability, performance and scalability over the Hadoop 2 environment. Yassine et al. [36] showed a knowledge-based IoT data analytics framework for smart homes. The configured fog nodes across cloud system perform effectively in data collection, processing and real-time recognition from the end-user activities. The overall results and research study employs beneficiary outcomes to establish real smart homes through orchestrated IoT networks.

Our proposed BDPS scheme follows depth-first search-based map reduced scheduling policies for high-speed packet deliveries at the network edges. The map reduced (i.e., shorten the active master–slave communication path or routing) scheduling mechanism supports the Spark-RDD in-memory data processing mechanism and carries the packet distribution across the mesh overlay network. The BDPS scheme provides efficient computation, and low latency around the fog-IoT network hop ends. A brief comparison is shown in (Table 2) to gain insight throughout the existing key technologies and research gaps.

Table 2. Emerging key technologies, contributions and research gaps.

Works	Key Technologies					Contributions and Research Gap
	Cloud	**Fog**	**IoT**	**Map.R**	**Big.D**	
Farjana et al. (2020) [2]	✓	✓	✓	X	X	Security issues declared in fog only services.
Ramya et al. (2012) [4]	✓	X	X	X	X	Dynamic graph query module for NaaS Cloud.
Oma et al. (2018) [23]	X	X	✓	X	X	WSN platform for serving lower fogs.
Tajalli et al. (2020) [27]	X	X	✓	X	X	DoS security aware framework for WSANs.
Forti et al. (2021) [28]	X	X	X	✓	X	Application placement for map reduced services.
Zhihui et al. (2017) [29]	X	X	✓	✓	✓	Weighted Linear Regression for hadoop clusters.
Swain et al. (2020) [30]	X	✓	✓	X	X	TDMA task offload scheduler for fog-IoT nodes.
Saito et al. (2020) [31]	X	X	✓	X	X	Wireless message subscription model for IoT nodes.
Vasudevan et al. (2020) [32]	X	X	✓	X	X	Scheduler problem in distributed IoT labeling.
Ortiz et al. (2022) [33]	✓	✓	✓	X	X	Inappropriate 3-tier interfacing for large data set.
Postoaca et al. (2020) [34]	✓	X	X	X	X	Distributed algorithmic access for cloud datacenter.
Yassine et al. (2018) [36]	✓	✓	✓	X	✓	Only big data based task schedulers.
Saba et al. (2021) [37]	X	✓	✓	X	X	Dynamic IoT traffic model for fog analytics.
Li et al. (2019) [38]	✓	✓	✓	X	X	Shortest paths algorithms for road networks.
Baranwal et al. (2021) [39]	✓	✓	✓	X	X	Data deliverable services and shared architectures.
Honar et al. (2021) [40]	✓	X	✓	X	✓	Facing trouble in middle ware latency support.
Bendechache et al. (2020) [41]	✓	✓	✓	X	X	Gap in middleware feasibilities.
Losada et al. (2021) [46]	✓	✓	✓	X	X	Gaps in between Proposed and ES scheme.

Map.R—declares map reduce, Big.D—declares big data in Table 2.

4. System Model and Methodology

In this section, we describe our system model and methodology.

4.1. BDPS Data Processing in Cloud, Fog and IoT Network Layer

We developed a high-speed data delivery mechanism to increase efficient data distribution at the IoT network end. The illustrated scheme enhances the data distribution pattern by working through in-memory Spark-RDD accelerator-based process schedulers across the fog-IoT network ends while performing distributed data sharing activities. The layered architecture is divided into three segments—cloud, fog, and IoT layers (e.g., end devices). The three-layer data distribution architecture (Figure 4) works in a bottom-up fashion. IoT network layers are the concerning part for major data loads to distribute toward the end devices. We contrived the in-memory Spark-RDD accelerator-based process schedulers (i.e., BDPS scheme) through a fog layer that works within the fog-IoT orchestration to reduce the data loads at the network end, specifically in the end devices. Here, the fog layer serves as a primary helper for the IoT and a secondary helper for the cloud end to overcome the extra data processing load of mesh networks over the IoTs by providing the utmost data delivery enhancement or service efficiency toward the end devices. The fog has Spark-RDD process schedulers (e.g., in-memory depth-first search accelerator) for the concerned IoTs or end devices. For data jamming, excess process overload in the IoT layer end, the fog carrier schedules the in-memory Spark-RDD accelerator toward the fog-IoT gateway connection and sends back the information to the cloud end for a process log list and further possible in-memory depth-first search (i.e., DFS) schedule verification. The fog layer has a SP (e.g., session password) timestamp to shut off the BDPS process to run after the regular processed data loads. The fog generates in-memory process schedulers while carrying the data after being overridden by the SP timestamp for scheduled data toward the IoT and central cloud layers containing the information of the in-memory-based process accelerations. Thus, the system's data acceleration improves with efficient process deliveries toward the network ends.

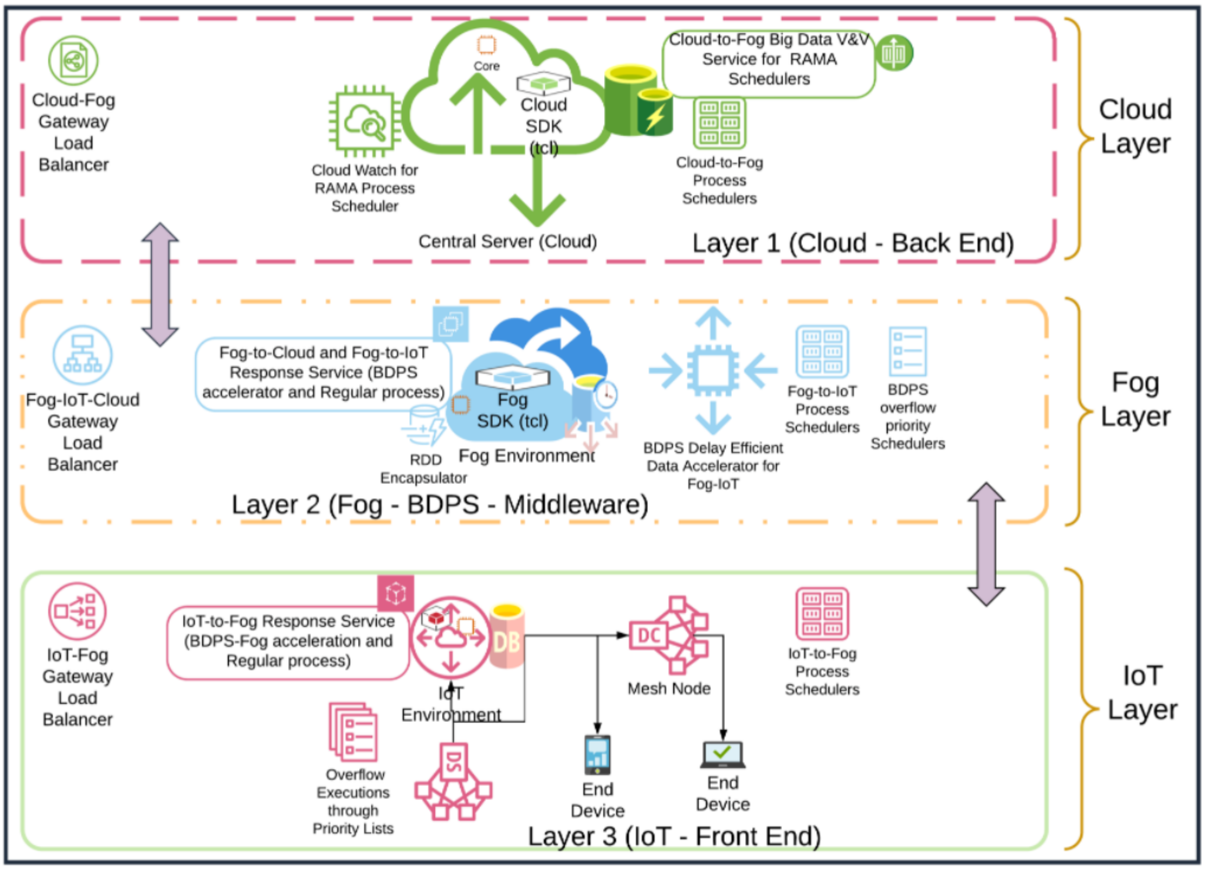
Cloud-Fog Gateway Load Balancer
Core
Cloud SDK (tcl)
Cloud-to-Fog Big Data V&V Service for RAMA Schedulers
Cloud Watch for RAMA Process Scheduler
Cloud-to-Fog Process Schedulers
Central Server (Cloud)
Layer 1 (Cloud - Back End)
Cloud Layer
Fog-IoT-Cloud Gateway Load Balancer
Fog-to-Cloud and Fog-to-IoT Response Service (BDPS accelerator and Regular process)
Fog SDK (tcl)
RDD Encapsulator
Fog Environment
BDPS Delay Efficient Data Accelerator for Fog-IoT
Fog-to-IoT Process Schedulers
BDPS overflow priority Schedulers
Layer 2 (Fog - BDPS - Middleware)
Fog Layer
IoT-Fog Gateway Load Balancer
IoT-to-Fog Response Service (BDPS-Fog acceleration and Regular process)
IoT Environment
DB
DC
Mesh Node
IoT-to-Fog Process Schedulers
Overflow Executions through Priority Lists
DS
End Device
End Device
Layer 3 (IoT - Front End)
IoT Layer

Figure 4. Overview of our developed BDPS scheme.

4.2. Components of the Fog-IoT Hierarchical Overlay Network and BDPS Architecture

Our system architecture is divided into three conventional layers: the back-end, middleware, and front-end services. The three-layer architecture may be categorized as (1) the cloud layer or core (back-end), (2) the fog layer (middleware), and (3) the IoT layer (front-end).

The core or cloud layer: The central cloud or service tracks the active participation of BDPS process schedulers to deliver from the fog toward the cloud in a secure fashion through the SP (session password). According to the process loads at the fog-IoT network ends, the cloud sends verification and validation to the fog to start the BDPS to reduce (Algorithm 1) the extra delay overheads. The current fog sessions work to process the data allocations assigned for the allocated devices. The overall cloud-to-fog communication creates an initialization delay (e.g., session-based configured counter for packet scheduling). The fog sessions resume the work through re-scheduling (i.e., enabling BDPS) for the previous information (e.g., overloaded information at the network ends) for creates extra jamming loads. The cloud comprises a data center that acts as a reservoir and contains the logs or verified proofs for the generated BDPS scheme timings (e.g., a counter for the affected processed packets). Later, the reservoir sends a code of conduct to the fog to cause a re-scheduling (i.e., in-memory-based Spark-RDD or BDPS) of synchronized processed packets within the fog-IoT orchestrations. Each renewed processed packet includes the information of the previous overloaded packets and the new super-seeded packets (i.e., BDPS-verified packets) as a hash table generation for the fog-IoT nodes. The hash table is a cloud-only mechanism or cloud-native function for assuring the process verification and validation. Thus, the developed scheme re-distributes the synchronized high-speed packets throughout the fog-IoT network ends with no loss of data with a delay efficient property.

The fog layer or middleware: The fog layer is equipped with a timer-based scheduled counter that acts as middleware for the cloud and IoT for super seeding (i.e., high-speed deliveries through in-memory) the extra overflow of processes or data jamming toward the IoT network ends. The fog layer establishes a Spark-RDD in-memory data accelerator (i.e., BDPS) mechanism to synchronize the processed data toward the IoT ends. The fog gathers direction from the cloud and verifies the needful (i.e., search for extra loaded data over packets using depth-first strategy) to generate the BDPS scheme for overcoming the extra user loads (i.e., data stress) across the fog-IoT ends. The BDPS scheme gathers the previous overloaded process stacks, aligns them with a counter track, and feeds them into the RAMA scheduler running or generated processes into the memory (i.e., RAM and cache for new and previous feeding sequentially) for the fast delivery of data distribution. The information-gathering procedure is performed regularly (Algorithm 1) after each process execution and verification from the cloud. The fog layer comprises a timestamp, SP (e.g., session password), which is generated if BDPS is needed to establish and track by the cloud reservoir. Furthermore, for the data load, the fog generates a counter-timer-based depth-first packet search strategy to encapsulate the loaded processes and check back for the cloud verification. After verification and validation, the fog layer starts the BDPS scheme, a mechanism for in-memory Spark-RDD data accelerator, to re-synchronize the packets (Algorithm 2) and deliver them to the respective device ends. Thus, the scheme synchronized past and newly created processes by supporting data loss avoidance by delivering it in a time-wise (i.e., high-speed process accelerators) function pattern. The fog layer also has a similar security pattern (i.e., SP) as the cloud to provide extra security for the packet schedulers.

The IoT layer or front-end service for the attached fog and cloud layer: The IoT layer has permission to connect with individual devices across the network for verification from the fog-IoT orchestration service. The IoT layer is concerned with data overloads and needs to check for the loaded process. The derived system design follows a generalized pattern for the fog-IoT layer to connect with different devices within the network. However, the architecture proceeds with the BDPS scheme only after initializing the session (Algorithm 2) from the fog through the cloud verification and validation. On the other hand, the scheme has an SP (session password) mechanism for secured data flow procedures. The SP mechanism (Table 3) provides secure packet distribution through BDPS over the fog, rescuing the cloud from packet sniffing problems.

Table 3. Symbols used in the algorithms.

Symbol	Definition
n_c	Current process packet
(n_c+1)	Increment of packet stacks to the next upcoming
$clss$	Cloud session service container
fss	Fog mapreduce-Spark service container
iss	IoT cluster session service
n	Processing packet at first start

Algorithm 1: BDPS—delay efficient scheme for rapid data delivery (cloud, fog to IoT end)—Part 1

```
Input: Overflow of queuing data processes at the network ends based on priority
       schedulers
Output: in-memory map-reduction-based functional RDD-RAMA data
        accelerators
/* isOverflow = A Boolean value defines data Overflow of processes
   or Not, denoted as $of$                                              */
/* PriorityPacketList = Priority-based threaded packet counter,
   denoted as $pl$                                                      */
/* SchedulerRAMA = Particular feed-forwarding scheduled packet for
   data into both RAMandCacheMemoryList, denoted as $srama$ and $rcml$
   */
/* Verification and Validation Info, denoted as $vvi$                   */
/* Fog Session Service, denoted as $fss$                                */
/* IoT Cluster Session Service, denoted as $icss$                       */
/* Cloud Session Service, denoted as $clss$                             */
foreach srama in rcml do
    Step 1: Actor: clss
        $vvi \leftarrow srama$.get $vvi$()
        if srama is NOT from regular packet execution AND vvi() generated then
            $of \leftarrow$ true
            $n \leftarrow pl$.size()
            $srama$.number $\leftarrow n + 1$
            send.$srama$ To $fss$
        end
        else
            $of \leftarrow$ false
            check.previous scheduled process queue info () of the $srama$
            Assign existing process delivery priority queues() To $fss$
        end
    if $of$ then
        Step 2: Actor: fss
            regenerate.RAMA scheduler()
            get.validated from Cloud
            reallocate.RAMA with running process schedulers()
            reduce.bottlenecks at IoT end()
            $n_c \leftarrow$ current executing process packet scheduler()
            traverse.($n_c$) To priority reallocation()
            $srama$.number $\leftarrow (n_c+1)$
    end
end
```

Algorithm 2: BDPS—delay efficient scheme for rapid data delivery (cloud, fog to IoT end)—Part 2

```
Input: Overflow of queuing data processes at the network ends based on priority
       schedulers
Output: in-memory map-reduction-based functional RDD-RAMA data
        accelerators
/* $isOverflow$ = A Boolean value defines data Overflow of processes
   or Not, denoted as $of$                                            */
/* $PriorityPacketList$ = Priority based threaded packet counter,
   denoted as $pl$                                                    */
/* $SchedulerRAMA$ = Particular feed-forwarding scheduled packet for
   data into both $RAMandCacheMemoryList$, denoted as $srama$ and $rcml$
   */
/* $Verification\ and\ Validation\ Info$, denoted as $vvi$            */
/* $Fog\ Session\ Service$, denoted as $fss$                          */
/* $IoT\ Cluster\ Session\ Service$, denoted as $icss$                */
/* $Cloud\ Session\ Service$, denoted as $clss$                       */
if $of$ then
    Step 2: Actor: $fss$
        Increment $srama$ Numbers By One()
        else
            Step 2: Actor: $fss$
                Assign existing and new scheduled processes() to IoT cluster ends
            Step 3: Actor: $icss$
                if Cloud and fog vvi is confirmed then
                    Continue the fog $srama$ processes()
                    $srama$.get allocated
                    numbers according to $pl$.size()
                    set.configured $srama$ to intended
                    IoT device()
                    Generate and process $srama$ counter() for cloud $vvi()$
                end
                else
                    Step 3: Actor: $icss$
                        get.regular process execution ()
                        from cloud–fog ends
                        set.delivery according
                        To $pl$.size()
                        Assign existing and new scheduled processes() to
                        IoT intended device ends
                end
        end
end
```

5. Experimental Research and Analysis

We compared the established scheme (BDPS) with the traditional Bellman–Ford (BF) algorithm, the Floyd–Warshall (FW) algorithm, Dijkstra algorithm (DA), and the Apache Hadoop (AH) algorithm. The algorithmic performance considerations were checked (Table 4) under these parametric data delivery (i.e., service response) metrics across the cloud, fog, and IoT continuum: packet drop (time), network node execution or network latency (time), frame delivery (time), network overhead (time), computational delay (time) and throughput (time). See Tables 5 and 6 for simulation environment details.

Table 4. Performance criteria and baselines.

Performance Criterion	**Baselines**
Throughput [Mb/s]	Cloud Orchestrated Fog-IoT network
Node Failure Rate [%]	Spark-RDD based Fog-IoT Controller Scheduling
Computational Delay [s]	Schedule-Aware Bundle Routing (SABR)
Bandwidth Prediction [Kb/s]	Preemptive Scheduling Regressor Spark-RDD Regressor Asymptotic Regressor

Table 5. Parameters of the simulation environment (software centric).

Simulation Parameters	**Values**
	Software Centric Parameters
Network emulator	OMNET++ 5.5.3; Jupyter Notebook 6.1.0; ns-3.2.1; iFogSim
Cloud storage platform	Owncloud 10.3
Packet analyzer	Wireshark Packet Analyzer
Programming language	Python 3.5.3, Java 9, C++17

Table 6. Parameters of the simulation environment (device centric).

Simulation Parameters	**Values**
	Cloud Parameters
No. of Fog controllers	1
No. of CloudBus switches	2
No. of Cloud gateways	1
Type of Cloud controllers	HP Z4 G4 Workstation
Cloud routing protocol	TCP/IP
	Fog Parameters
No. of Fog controllers	4
No. of FogBus switches	6
No. of Fog gateways	4
Type of Fog controllers	Dell Optiplex 3060 MT; Raspberry Pi 3b+
Fog routing protocol	TCP/IP
	IoT Parameters
Mobility model	Reference point group model (RPGM)
Type of IoT Controller	Amazon Fire HD 7; Dell Edge Gateway 5000
Traffic type	Constant Bit Rate (CBR)
Number of IoT devices	100
Simulation time	600 s
Simulation area	5000 m $\times$ 5000 m
Data rate	20 Mbps
Transmitted packet size	256–1024 B
Initial energy value	10–12 J
Initial trust value	5 J

5.1. Throughput Generation Scenario over the FW, AH, BF, DA and BDPS Algorithm in Cloud, Fog, IoT Orchestration

Figure 5 illustrates that the Bellman–Ford (BF) algorithm has a higher throughput in the fog continuum compared to IoT and cloud. BF does not employ data mapping toward intended users. The data offloads are higher in the IoT, and cloud redirects the middleware inconsistency of data processing within the three-tier architecture. BF declares a higher throughput of 10 ms in fog, showing a multiplication of data packet offload of 2.4% in 1000 packet deliveries. However, having a resilient map reduction data architecture, the

BDPS shows higher throughputs of 10 ms in fog, 8 ms in IoT and 7 ms in cloud, respectively, and the data offload remains 1.4% in 1000 packet deliveries.

This data execution pattern of BDPS shows consistency within the three-tier data distribution architecture across intended nodes. Less data offload resembles better data deliveries across the fog-IoT network continuum. The FW, DA and AH algorithms perform the same in terms of data offload around the cloud, fog, and IoT orchestration, and employ lower data throughput. Comparing to BF and others, the BDPS performs better, delivering the map reduction data delivery pattern toward the network edges.

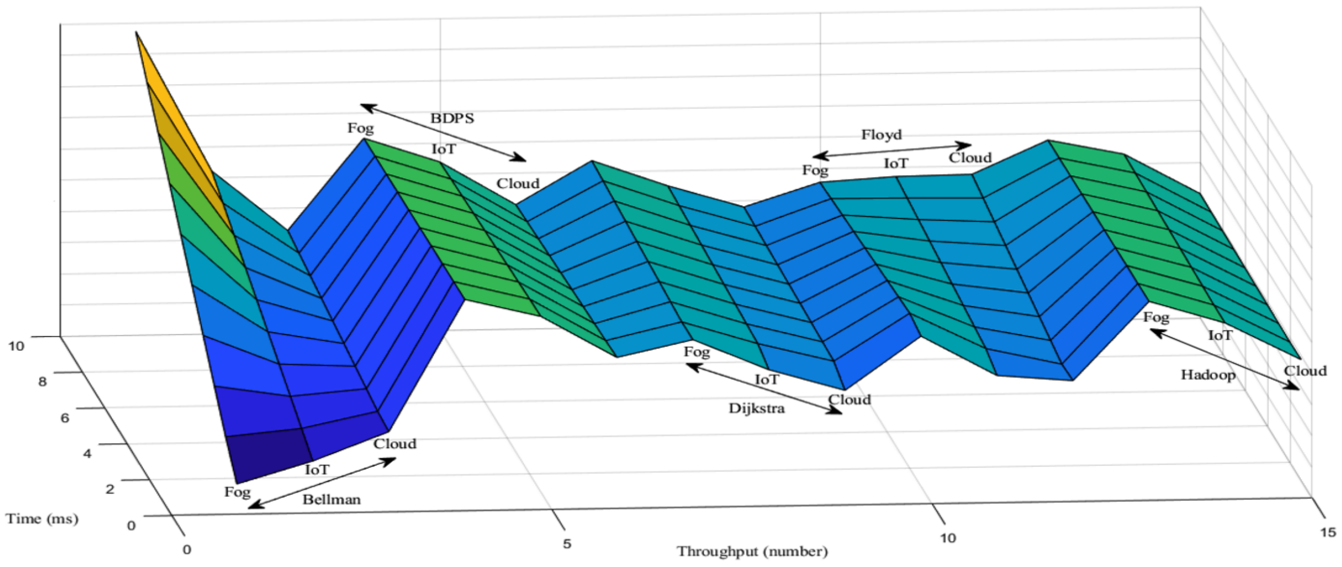

Figure 5. Throughput of data execution among existing algorithms and BDPS over fog-IoT mesh architecture.

5.2. Network Overhead Generation Scenario over the FW, AH, BF, DA and BDPS Algorithms in Cloud, Fog, and IoT Orchestration

Figure 6 describes that the Bellman–Ford (BF) algorithm has a higher network overhead in the fog continuum rather than cloud and IoT. BF does not employ data mapping toward intended users. The data offloads are higher in the fog, and cloud (rather than IoT) redirects the stack overload of data processing within the three-tier mesh architecture. Hence, BF, DA, FW and AH are not a worthy choice for the increased network overhead over system run time. BF declares a higher network overhead of 12 ms in fog, showing a multiplication of the data packet offload of 1.6% in 1000 packet deliveries. However, having a resilient map reduction data architecture, the BDPS shows lower network overhead of 7 ms in fog, 6.5 ms in cloud and 5.9 ms in IoT, respectively, and the data offload remains 0.7% in 1000 packet deliveries. This data execution pattern of BDPS shows gradual consistency within the three-tier data distribution architecture across intended nodes. Less network overhead resembles better data deliveries across the fog-IoT network continuum. The FW, DA and AH algorithms show a gradual increase in the data offload similar to BF throughout the cloud, fog, and IoT orchestration, and employ higher data offload. Comparing to BF and others, in terms of network overhead, BDPS performs better, delivering the map reduction data delivery pattern toward the network edges.

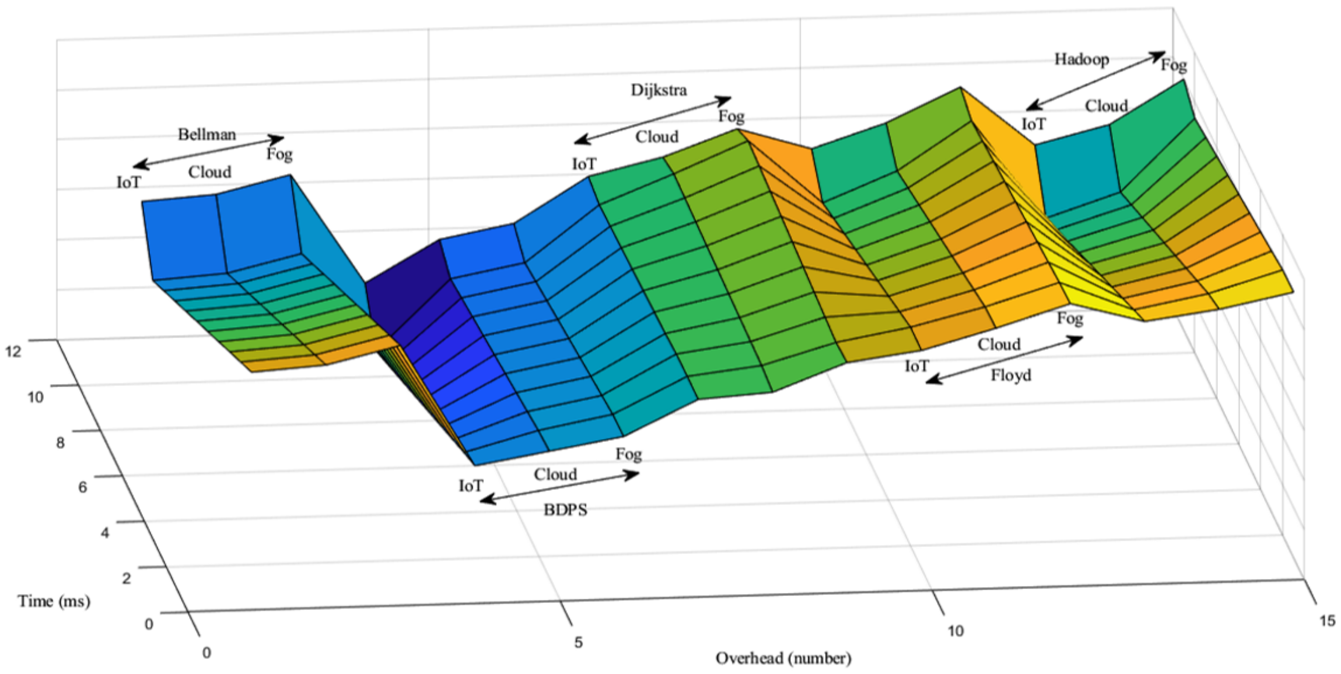

Figure 6. Network overhead among existing algorithms and BDPS over fog-IoT mesh architecture.

5.3. Packet Drop Generation Scenario over the FW, AH, BF, DA and BDPS Algorithms in Cloud, Fog, and IoT Orchestration

Figure 7 shows that the Bellman–Ford (BF) algorithm has a higher packet drop in the cloud continuum rather than IoT and fog. The BF does not employ data mapping toward intended users. The data hand-offs or packet drop are higher in the cloud, and IoT (rather than fog) redirects data stack overloads within the three-tier mesh architecture. Hence, BF, DA, FW and AH are not a worthy choice for increased packet drops over the system run time. BF declares a higher packet drop of 12 ms in fog, showing a multiplication of the packet offload of 2.2% in 1000 packet deliveries. However, having a resilient map reduction data architecture, BDPS shows lower packet drops of 11 ms in fog, 10 ms in IoT and 12 ms in cloud, respectively, and the packet drop or data hand-off remains at 0.2–0.3%, respectively, in 1000 packet deliveries. This data execution pattern of BDPS shows gradual consistency within the three-tier data distribution architecture across intended nodes. Less packet drop resembles better data deliveries across the fog-IoT network continuum. The FW, DA and AH algorithms show a gradual increase in packet drop similar to BF throughout the cloud, fog, and IoT orchestration, and employ higher data hand-off. Comparing to BF and others, in terms of packet drop, BDPS performs better, delivering the map reduction data delivery pattern toward the network edges.

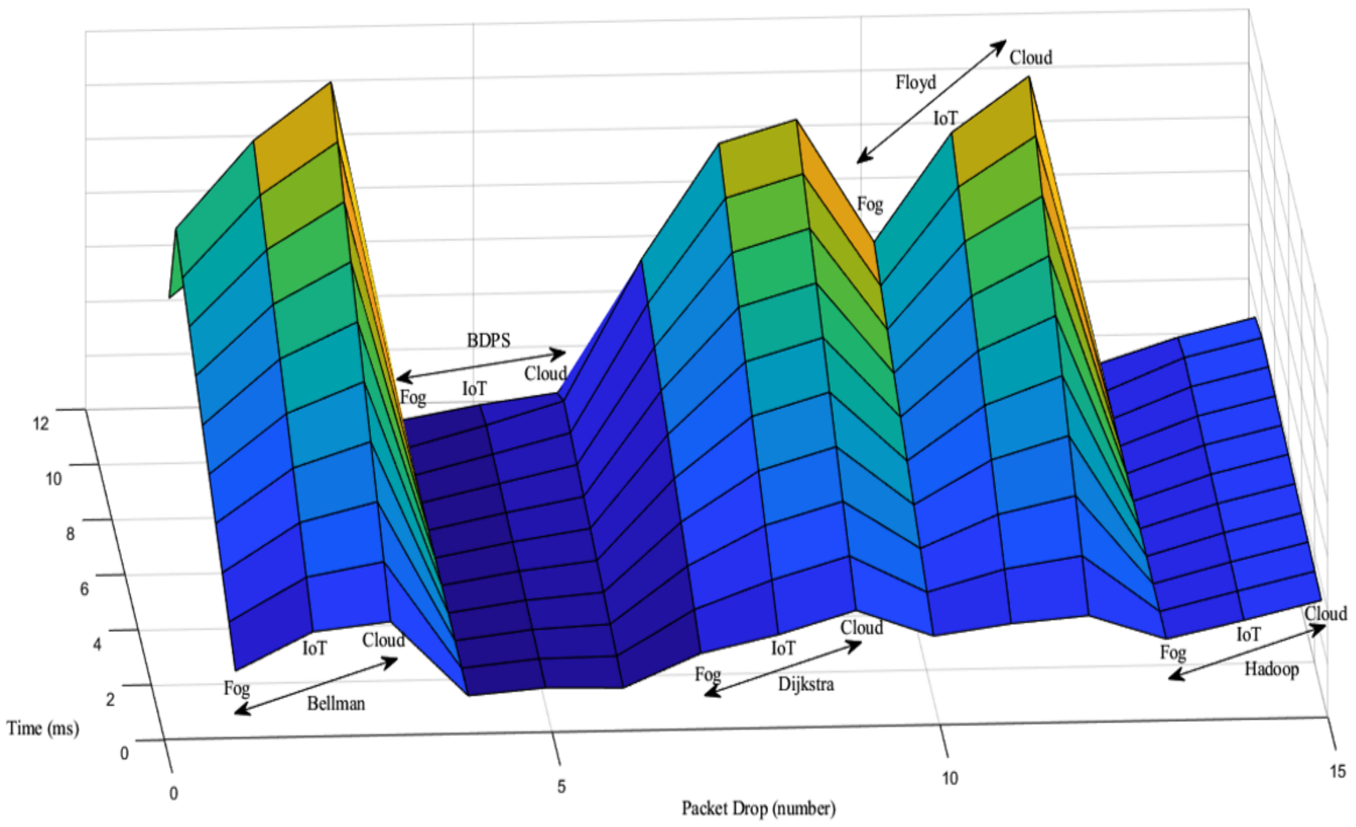

Figure 7. Packet drop generation scenario over the FW, AH, BF, DA and BDPS algorithms in cloud, fog and IoT orchestration.

5.4. Network Efficiency Generation Scenario over the FW, AH, BF, DA and BDPS Algorithms in Cloud, Fog, and IoT Orchestration

Figure 8 depicts that the Bellman–Ford (BF) algorithm has better efficiency in the fog continuum rather than IoT and cloud. BF does not employ data mapping toward intended users. The packet hand-offs are higher in the IoT, and cloud redirects the middleware inconsistency of data processing within the three-tier architecture and fog continuum misplaced stack handlers. BF declares higher efficiency of 12 ms in fog, showing a multiplication of misplaced stack handling of 2.5% and faster process execution in 1000 packet deliveries. However, having a resilient map reduction data architecture, BDPS shows better efficiency, without misplaced data handling, of 12 ms in fog, 10 ms in IoT and 7 ms in cloud, respectively. The misplaced packet handling remains at 1.2% in 1000 packet deliveries.

This data execution pattern of BDPS shows consistency within the three-tier data distribution architecture across fog-IoT intended nodes. Less misplaced packet handling resembles better data deliveries across the fog-IoT network continuum among other algorithms, namely FW, DA and AH. FW, DA and AH algorithms show less misplaced data handling around the cloud, fog, and IoT orchestration but cannot deliver resilient data mapping toward the intended edge nodes. Comparing to BF and others, BDPS performs better, delivering map-reduction data delivery patterns toward the network edges.

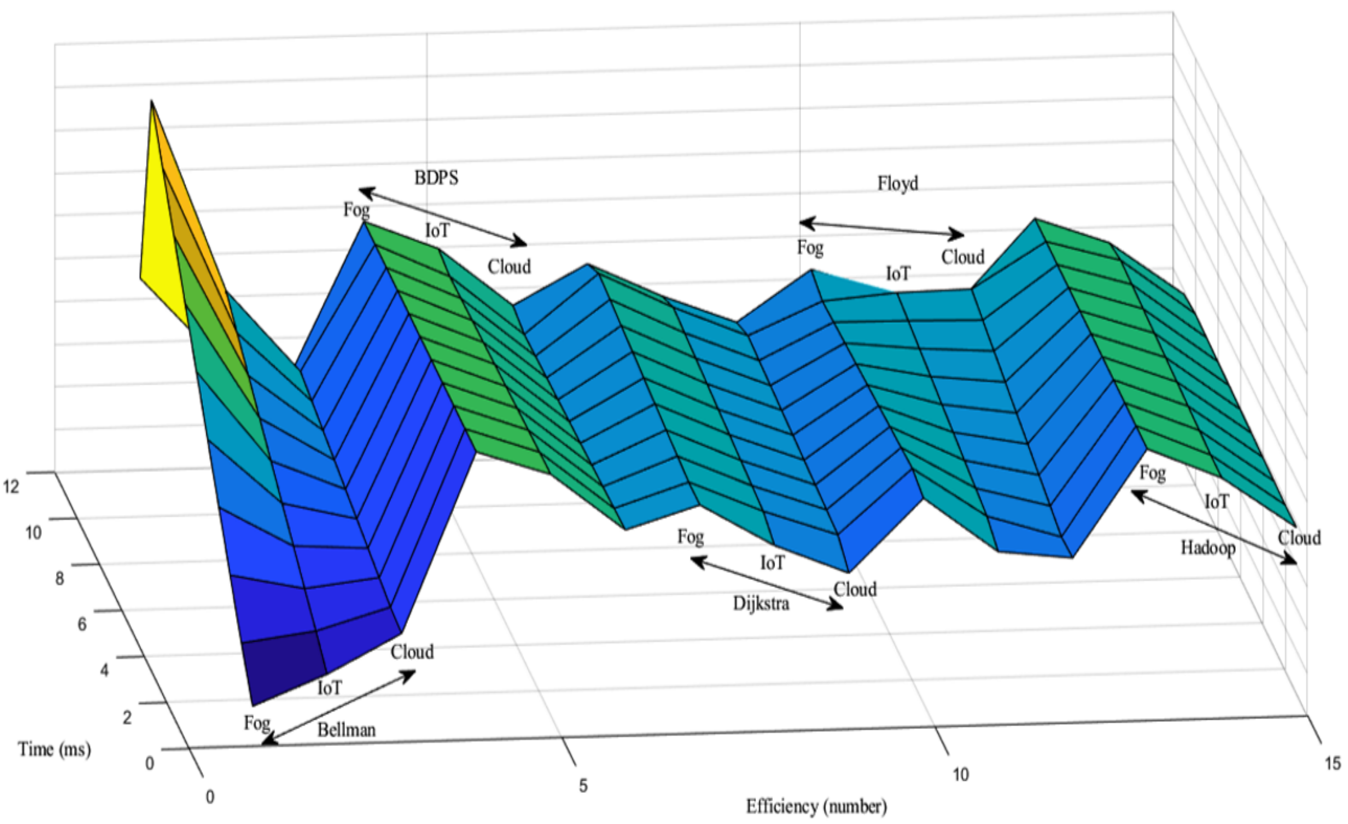

Figure 8. Network efficiency generation scenario over the FW, AH, BF, DA and BDPS algorithms in cloud, fog, and IoT orchestration.

5.5. Computational Delay Generation Scenario over the FW, AH, BF, DA and BDPS Algorithms in Cloud, Fog, and IoT Orchestration

Figures 9 and 10 show that the Dijkstra algorithm (DA) has a higher computational delay in the fog-IoT continuum and does not employ data mapping toward edge network intended users. The data hand-offs and packet delays across the edge nodes are higher, which redirects the inconsistency of data processing within the three-tier mesh architecture. The Dijkstra algorithm declares an overall higher computational delay of 45 ms in 40 edge node data executions. Having a resilient map reduction data architecture, the BDPS shows an overall lower computational delay of 2 ms in 3 edge node execution and 26 ms overall computational delay for 40 edge node execution. The Bellman–Ford, Floyd–Warshall and Apache Hadoop algorithms show 66 ms, 85 ms and 80 ms, respectively, for 40 edge node data execution.

This data execution pattern of BDPS shows consistency within the three-tier data distribution architecture across intended nodes. Less computational delay from the box plot statistics resembles better data deliveries across the fog-IoT network continuum. The histogram plot shows that we have maximum edge node data delivery in 5–40 ms. That supports our BDPS higher data delivery capacity across the edge nodes. Considering the FW, DA, AH and BF algorithms, in terms of computational delay and having a resilient data delivery architecture, BDPS is a better choice for fog-IoT-based data orchestration. However, comparing with computational delay and box plot statistics, we can derive that BDPS performs better compared to the others. Hence, BDPS is a desirable and beneficial scheme for the fog-IoT hierarchical overlay mesh architecture.

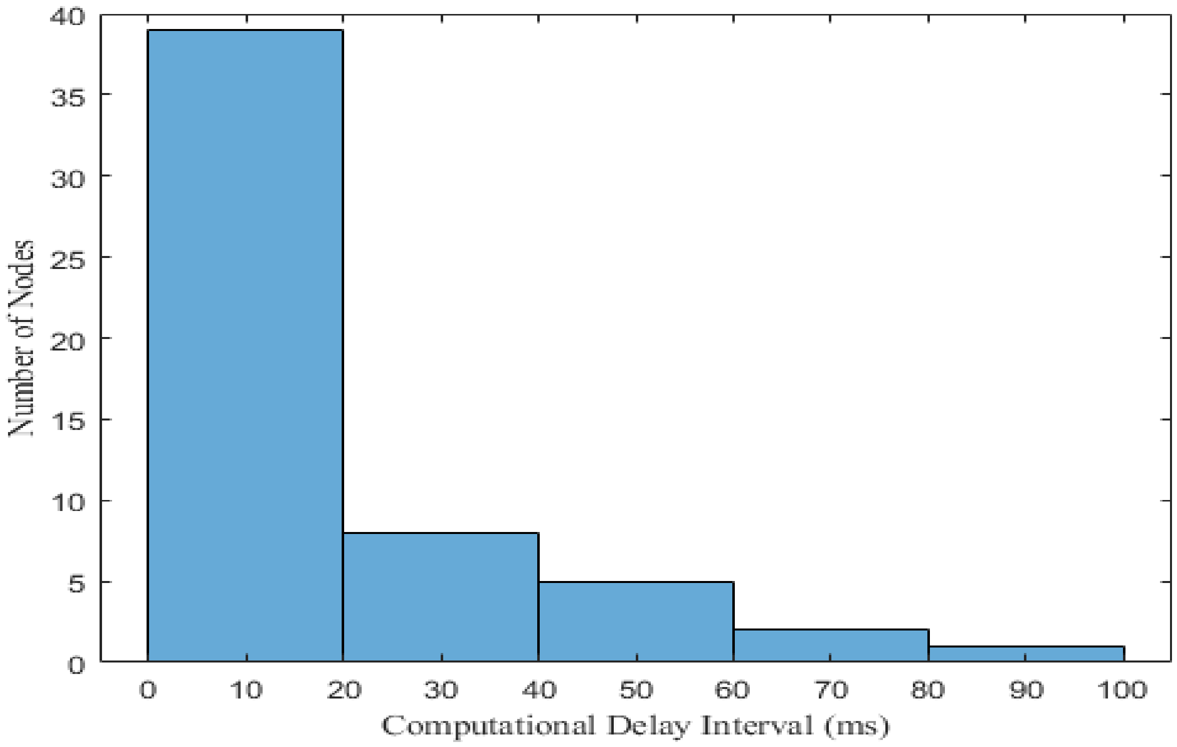

Figure 9. Histogram view of computational delay over node execution among existing algorithms and BDPS.

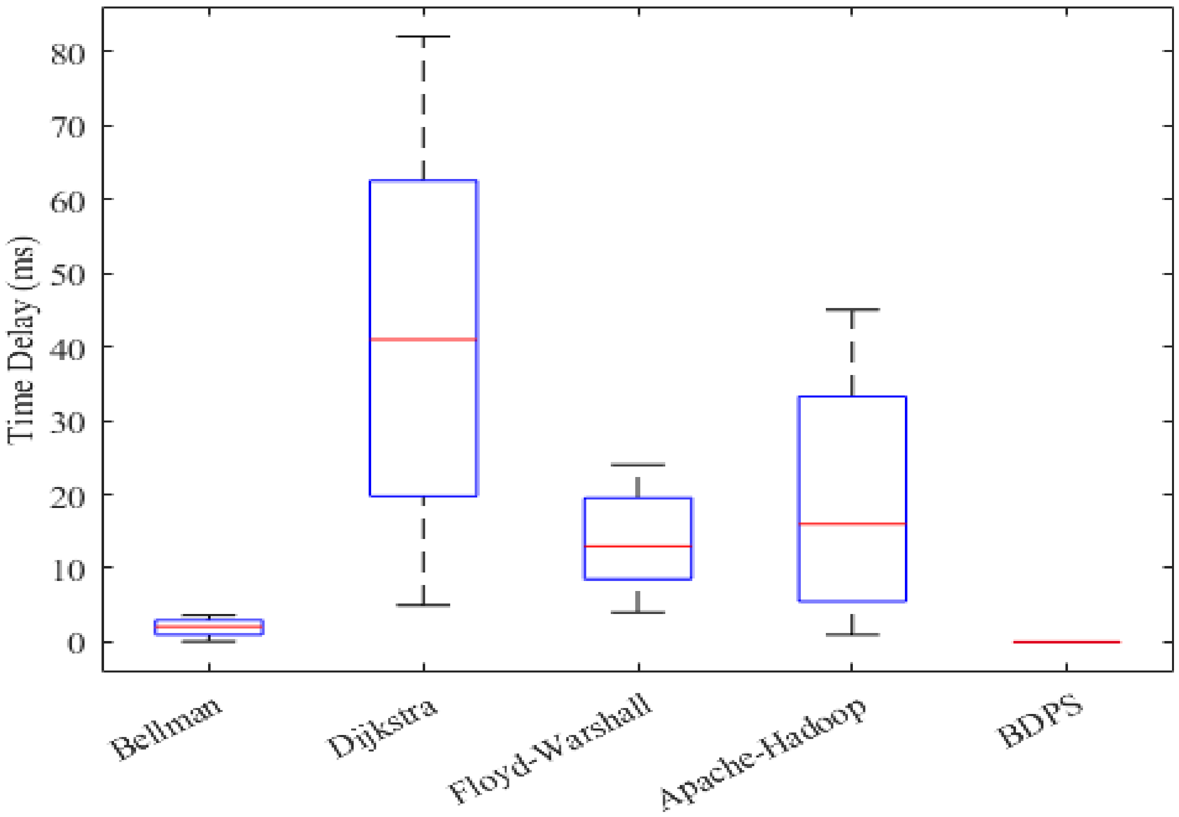

Figure 10. Box plot statistics of time delay among existing algorithms and BDPS.

6. Discussion

In this work, we employed a scheme to ensure rapid data delivery within the fog-IoT network through a Spark-RDD in-memory processing-based accelerator scheme (i.e., BDPS). We implemented a heterogeneous hierarchical-tree-based mesh overlay network to display the process executions within the distributed fog-IoT orchestrations scenario with the BDPS scheme. In the IoT layer, if a data overflow occurs at the end of an IoT device, the running information in the processing frame is scheduled for verification and validation to get feed-forwarded through BDPS. Our configured scheme acts through timer-based session password generation (SP) from fog to IoT and fog to cloud in terms of data execution. Hence, our scheme is developed as a three-layer architecture, where the central cloud server acts as a hidden backbone (i.e., reservoir) for the log file verification and validation. The central cloud processes the operations by collecting from the fog session, giving the command to regenerate the queries for validation, and starting the BDPS scheme toward fog-IoT orchestration deliveries. The newly processed stack (i.e., overloaded encapsulate packets) from the fogs are assured, compiled, and delivered across the IoT bases. The fog cluster generates the BDPS process only if there occurs an overflow of process jamming or bottleneck across the fog-IoT network ends. The in-memory Spark-RDD rapid data delivery mechanism supports a high-speed data execution procedure by encapsulating previous jammed packets and distributing them to their intended authority (i.e., dwelled devices) around the fog-IoT network mediums. Thus, the BDPS scheme provides less network latency and delays, resulting in efficient process deliveries across the connected network ends. The individual devices also have a hashing agreement to collect, process, and deliver data toward their respective ends. Thus, the occupied data-link layer safety for less data loss is assured and verified. The SP of each device puts a record over the cloud end if it goes through the BDPS scheme verification of rapid data transfer agreements. Our established BDPS scheme delivers an in-memory RAMA data scheduling and processing strategy that is efficient in providing less delay time over data executions at the hop-to-hop ends.

In our new algorithmic scheme—BDPS—the over-flooded IoT network nodes have new hashing sessions of information (e.g., previous and encapsulated ones). The hashing table of the fog-IoT nodes is carried upon packet overflow in a data table. Thus, the BDPS scheme is verified to feedforward the over-flooded packets and convert them into bigger packets while delivering them rapidly within a prioritized manner. In this architecture, the client sides (e.g., IoT and fog) rely on the BDPS scheme, whereas the server-side (e.g., cloud) acts only in a needed situation to verify the scheme. In the data overflow situation, the fog node carries the alert and delivers it via cloud verification to enable BDPS needed for the desired nodes or devices. The intermediate node carriers (i.e., devices in awaiting pipeline for BDPS) act according to their prioritized packets, and hence, deliver themselves for the respective BDPS feed-forwarding data acceleration process within the fog-IoT network orchestrations. The established BDPS scheme continues to generate the number of prioritizing packet schedulers iteratively until the data bottleneck occurs across the desired network ends.

The BDPS scheme performs well considering other existing algorithms. However, for the further development of this scheme, we will focus on comparatively large scaled-up architecture and algorithmic complexity issues in future.

7. Conclusions

It is difficult to maintain an efficient process delivery inside a cloud-based heterogeneous fog-IoT orchestration. We created a delay-efficient process delivery strategy (i.e., BDPS) for a fog-IoT overlay tree-based hierarchical mesh architecture in this study. The performance of our delay efficient algorithmic approach, BDPS, was superior to the BF, DA, FW, and AH algorithms. The inflated BDPS method suppresses the BF, DA, FW, and AH algorithms. The overall BDPS scheme is three times more efficient in data delivery across the fog-IoT coordination. Generally, the BDPS scheme outperforms by 28% efficiency in node execution, compared to BF, DA FW, and AH. The result employs the beneficiary outcome of the BDPS, proving the low latency of existing algorithms at almost one-fourth of DA, half of BF, and one-third of FW and AH, respectively. The histogram and box plot statistics shows the beneficiary outcome of the BDPS scheme, proving its low latency of 26 ms rather than those of other existing algorithms, redirecting 45 ms in DA, 66 ms in BF, 85 ms in FW and 80 ms in AH, respectively, while running the central cloud network as a backbone service.

Author Contributions: Data curation, N.F. and M.S.; funding acquisition, M.W. and A.B.; investigation, R.H., M.W. and M.J.N.M.; methodology, R.H., M.J.I. and M.J.N.M.; project administration, M.W.; software, M.J.N.M.; supervision, M.W., M.N.U. and A.B.; visualization, N.F.; writing—original draft, R.H., M.N.U., M.J.I. and M.J.N.M.; writing—review and editing, M.W. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded through the "ICT Innovation Fund (2016-17): ICT Division, Bangladesh" and also partially supported through the Australian Research Council Discovery Project: DP190100314.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hossain, M.R.; Whaiduzzaman, M.; Barros, A.; Tuly, S.R.; Mahi, M.J.N.; Roy, S.; Fidge, C.; Buyya, R. A scheduling-based dynamic fog computing framework for augmenting resource utilization. *Simul. Model. Pract. Theory* **2021**, *111*, 102336. [CrossRef]
2. Farjana, N.; Roy, S.; Mahi, M.J.N.; Whaiduzzaman, M. An identity-based encryption scheme for data security in fog computing. In *Proceedings of International Joint Conference on Computational Intelligence*; Springer: Singapore, 2020; pp. 215–226.
3. Whaiduzzaman, M.; Gani, A.; Naveed, A. Towards enhancing resource scarce cloudlet performance in mobile cloud computing. *Comput. Sci. Inf. Technol.* **2015**, *5*, 1.
4. Raghavendra, R.; Lobo, J.; Lee, K.W. Dynamic graph query primitives for sdn-based cloudnetwork management. In Proceedings of the First Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13 August 2012; pp. 97–102.
5. Whaiduzzaman, M.; Naveed, A.; Gani, A. MobiCoRE: Mobile device based cloudlet resource enhancement for optimal task response. *IEEE Trans. Serv. Comput.* **2016**, *11*, 144–154. [CrossRef]
6. Mahi, M.J.N.; Hossain, K.M.; Biswas, M.; Whaiduzzaman, M. SENTRAC: A Novel Real Time Sentiment Analysis Approach Through Twitter Cloud Environment. In *Advances in Electrical and Computer Technologies*; Springer: Singapore, 2020; pp. 21–23.
7. EL-Garoui, L.; Pierre, S.; Chamberland, S. A New SDN-Based Routing Protocol for Improving Delay in Smart City Environments. *Smart Cities* **2020**, *3*, 1004–1021. [CrossRef]
8. Firouzi, F.; Farahani, B.; Marinšek, A. The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT). *Inf. Syst.* **2021**, 101840, in press. [CrossRef]
9. Whaiduzzaman, M.; Mahi, M.J.N.; Barros, A.; Khalil, M.I.; Fidge, C.; Buyya, R. BFIM: Performance Measurement of a Blockchain Based Hierarchical Tree Layered Fog-IoT Microservice Architecture. *IEEE Access* **2021**, *9*, 106655–106674. [CrossRef]
10. Manogaran, G.; Lopez, D.; Chilamkurti, N. In-Mapper combiner based MapReduce algorithm for processing of big climate data. *Future Gener. Comput. Syst.* **2018**, *86*, 433–445. [CrossRef]
11. Ragaventhiran, J.; Kavithadevi, M.K. Map-optimize-reduce: CAN tree assisted FP-growth algorithm for clusters based FP mining on Hadoop. *Future Gener. Comput. Syst.* **2020**, *103*, 111–122. [CrossRef]
12. Whaiduzzaman, M.; Barros, A.; Shovon, A.R.; Hossain, M.R.; Fidge, C. A Resilient Fog-IoT Framework for Seamless Microservice Execution. In Proceedings of the IEEE International Conference on Services Computing (SCC), Chicago, IL, USA, 5–10 September 2021; pp. 213–221.
13. Awan, M.J.; Farooq, U.; Babar, H.M.A.; Yasin, A.; Nobanee, H.; Hussain, M.; Hakeem, O.; Zain, A.M. Real-time DDoS attack detection system using big data approach. *Sustainability* **2021**, *13*, 10743. [CrossRef]
14. Whaiduzzaman, M.; Farjana, N.; Barros, A.; Mahi, M.; Nayeen, J.; Satu, M.; Roy, S.; Fidge, C. HIBAF: A data security scheme for fog computing. *J. High Speed Netw.* **2021**, *27*, 381–402. [CrossRef]
15. Adoni, W.Y.H.; Nahhal, T.; Aghezzaf, B.; Elbyed, A. The MapReduce-based approach to improve the shortest path computation in large-scale road networks: The case of A* algorithm. *J. Big Data* **2018**, *5*, 1–24. [CrossRef]
16. Quasim, M.T. Resource Management and Task Scheduling for IoT using Mobile Edge Computing. *Wirel. Pers. Commun.* **2021**, 1–18. [CrossRef]

17. Zhang, D.; Zhang, W.; Yang, R.; Guo, M.; Chen, C.M. A distributed computation of the shortest path in large-scale road network. *J. Ambient. Intell. Humaniz. Comput.* **2019**, 1–16. [CrossRef]
18. Alazzam, H.; AbuAlghanam, O.; Sharieh, A. Best path in mountain environment based on parallel A* algorithm and Apache Spark. *J. Supercomput.* **2021**, 1–20. [CrossRef]
19. Aslanpour, M.S.; Gill, S.S.; Toosi, A.N. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet Things* **2020**, *12*, 100273. [CrossRef]
20. Whaiduzzaman, M.; Sookhak, M.; Gani, A.; Buyya, R. A survey on vehicular cloud computing. *J. Netw. Comput. Appl.* **2014**, *40*, 325–344. [CrossRef]
21. Eswaran, S.P.; Sripurushottama, S.; Jain, M. Multi criteria decision making (mcdm) based spectrum moderator for fog-assisted internet of things. *Procedia Comput. Sci.* **2018**, *134*, 399–406. [CrossRef]
22. Moertini, V.S.; Adithia, M.T. Uncovering Active Communities from Directed Graphs on Distributed Spark Frameworks, Case Study: Twitter Data. *Big Data Cogn. Comput.* **2021**, *5*, 46. [CrossRef]
23. Oma, R.; Nakamura, S.; Duolikun, D.; Enokido, T.; Takizawa, M. Fault-tolerant fog computing models in the IoT. In Proceedings of the 13th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Taichung, Taiwan, 27–29 October 2018; pp. 14–25.
24. Abdel-Basset, M.; Mohamed, R.; Elhoseny, M.; Bashir, A.K.; Jolfaei, A.; Kumar, N. Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications. *IEEE Trans. Ind. Inform.* **2020**, *17*, 5068–5076. [CrossRef]
25. Huang, W.; Zhou, J.; Zhang, D. On-the-Fly Fusion of Remotely-Sensed Big Data Using an Elastic Computing Paradigm with a Containerized Spark Engine on Kubernetes. *Sensors* **2021**, *21*, 2971. [CrossRef]
26. Whaiduzzaman, M.; Ismail Sumi, A.; Barros, A.; Satu, M.S.; Razon Hossain, M. Towards Latency Aware Emerging Technology for Internet of Vehicles. In Proceedings of the 25th Pacific Asia Conference on Information Systems (PACIS), Dubai, United Arab Emirates, 12–14 July 2021.
27. Tajalli, S.Z.; Mardaneh, M.; Taherian-Fard, E.; Izadian, A.; Kavousi-Fard, A.; Dabbaghjamanesh, M.; Niknam, T. DoS-resilient distributed optimal scheduling in a fog supporting IIoT-based smart microgrid. *IEEE Trans. Ind. Appl.* **2020**, *56*, 2968–2977. [CrossRef]
28. Forti, S.; Gaglianese, M.; Brogi, A. Lightweight self-organising distributed monitoring of Fog infrastructures. *Future Gener. Comput. Syst.* **2021**, *114*, 605–618. [CrossRef]
29. Lu, Z.; Wang, N.; Wu, J.; Qiu, M. IoTDeM: An IoT Big Data-oriented MapReduce performance prediction extended model in multiple edge clouds. *J. Parallel Distrib. Comput.* **2018**, *118*, 316–327. [CrossRef]
30. Swain, C.; Sahoo, M.N.; Satpathy, A.; Muhammad, K.; Bakshi, S.; Rodrigues, J.J.; de Albuquerque, V.H.C. METO: Matching Theory Based Efficient Task Offloading in IoT-Fog Interconnection Networks. *IEEE Internet Things J.* **2021**, *8*, 12705–12715. [CrossRef]
31. Saito, T.; Nakamura, S.; Enokido, T.; Takizawa, M. August. Topic-based processing protocol in a mobile fog computing model. In Proceedings of the 23rd International Conference on Network-Based Information Systems (NBiS-2020), Victoria, BC, Canada, 31 August–2 September 2020; pp. 43–53.
32. Vijayalakshmi, R.; Vasudevan, V.; Kadry, S.; Lakshmana Kumar, R. Optimization of makespan and resource utilization in the fog computing environment through task scheduling algorithm. *Int. J. Wavelets Multiresolut. Inf. Process.* **2020**, *18*, 1941025. [CrossRef]
33. Ortiz, G.; Zouai, M.; Kazar, O.; Garcia-de-Prado, A.; Boubeta-Puig, J. Atmosphere: Context and situational-aware collaborative IoT architecture for edge-fog-cloud computing. *Comput. Stand. Interfaces* **2022**, *79*, 103550. [CrossRef]
34. Postoaca, A.V.; Negru, C.; Pop, F. Deadline-aware Scheduling in Cloud-Fog-Edge Systems. In Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), Melbourne, Australia, 11–14 May 2020; pp. 691–698.
35. Moura, J.; Hutchison, D. Fog computing systems: State of the art, research issues and future trends, with a focus on resilience. *J. Netw. Comput. Appl.* **2020**, *169*, 102784. [CrossRef]
36. Yassine, A.; Singh, S.; Hossain, M.S.; Muhammad, G. IoT big data analytics for smart homes with fog and cloud computing. *Future Gener. Comput. Syst.* **2019**, *91*, 563–573. [CrossRef]
37. Saba, U.K.; ul Islam, S.; Ijaz, H.; Rodrigues, J.J.; Gani, A.; Munir, K. Planning Fog networks for time-critical IoT requests. *Comput. Commun.* **2021**, *172*, 75–83. [CrossRef]
38. Li, L.; Guo, M.; Ma, L.; Mao, H.; Guan, Q. Online workload allocation via fog-fog-cloud cooperation to reduce IoT task service delay. *Sensors* **2019**, *19*, 3830. [CrossRef]
39. Baranwal, G.; Vidyarthi, D.P. FONS: A fog orchestrator node selection model to improve application placement in fog computing. *J. Supercomput.* **2021**, *77*, 10562–10589. [CrossRef]
40. Honar Pajooh, H.; Rashid, M.A.; Alam, F.; Demidenko, S. IoT Big Data provenance scheme using blockchain on Hadoop ecosystem. *J. Big Data* **2021**, *8*, 1–26. [CrossRef]
41. Bendechache, M.; Svorobej, S.; Takako Endo, P.; Lynn, T. Simulating resource management across the cloud-to-thing continuum: A survey and future directions. *Future Internet* **2020**, *12*, 95. [CrossRef]
42. Markakis, E.K.; Karras, K.; Sideris, A.; Alexiou, G.; Pallis, E. Computing, caching, and communication at the edge: The cornerstone for building a versatile 5G ecosystem. *IEEE Commun. Mag.* **2017**, *55*, 152–157. [CrossRef]
43. Wang, J.; Li, D. Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing. *Sensors* **2019**, *19*, 1023. [CrossRef] [PubMed]

44. Niu, X.; Shao, S.; Xin, C.; Zhou, J.; Guo, S.; Chen, X.; Qi, F. Workload allocation mechanism for minimum service delay in edge computing-based power Internet of Things. *IEEE Access* **2019**, *7*, 83771–83784. [CrossRef]
45. Ali, B.; Pasha, M.A.; ul Islam, S.; Song, H.; Buyya, R. A Volunteer-Supported Fog Computing Environment for Delay-Sensitive IoT Applications. *IEEE Internet Things J.* **2020**, *8*, 3822–3830. [CrossRef]
46. Losada, M.; Cortés, A.; Irizar, A.; Cejudo, J.; Pérez, A. A Flexible Fog Computing Design for Low-Power Consumption and Low Latency Applications. *Electronics* **2021**, *10*, 57. [CrossRef]
47. Taherizadeh, S.; Apostolou, D.; Verginadis, Y.; Grobelnik, M.; Mentzas, G. A Semantic Model for Interchangeable Microservices in Cloud Continuum Computing. *Information* **2021**, *12*, 40. [CrossRef]
48. Rocha Neto, A.; Silva, T.P.; Batista, T.; Delicato, F.C.; Pires, P.F.; Lopes, F. Leveraging edge intelligence for video analytics in smart city applications. *Information* **2021**, *12*, 14. [CrossRef]