applied sciences

MDPI

Article

Sonum: Software-Defined Synergetic Sampling Approach and Optimal Network Utilization Mechanism for Long Flow in a Data Center Network

Lizhuang Tan [1,2,*], Wei Su [1,2,*], Peng Cheng [3], Liangyu Jiao [4] and Zhiyong Gai [5]

1 School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing 100044, China
2 National Engineering Laboratory of Next Generation Internet Interconnection Devices, Beijing Jiaotong University, Beijing 100044, China
3 College of Computer and Communication Engineering, China University of Petroleum, Qingdao 266580, China; z17070643@s.upc.edu.cn
4 School of Information Engineering, SouthWest University of Science and Technology, Mianyang 621010, China; liangyu0112@126.com
5 School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China; gzhiyong94@bupt.edu.cn
* Correspondence: lzhtan@bjtu.edu.cn (L.T.); wsu@bjtu.edu.cn (W.S.)

Received: 14 November 2019; Accepted: 20 December 2019; Published: 24 December 2019

Featured Application: Compared to traditional ECMP and Hedera, Sonum can significantly improve network throughput, flow completion time and link utilization. Sonum is expected to be used in data center network management with the result of increasing data center revenue and improving user experience.

Abstract: Long flow detection and load balancing are crucial techniques for data center running and management. However, both of them have been independently studied in previous studies. In this paper, we propose a complete solution called Sonum, which can complete long flow detection and scheduling at the same time. Sonum consists of a software-defined synergetic sampling approach and an optimal network utilization mechanism. Sonum detects long flows through consolidating and processing sampling information from multiple switches. Compared with the existing prime solution, the missed detection rate of Sonum is reduced by 2.3%–5.1%. After obtaining the long flow information, Sonum minimizes the potential packet loss rate as the optimization target and then translates load balancing into an optimization problem of arranging a minimum packet loss path for long flows. This paper also introduces a heuristic algorithm for solving this optimization problem. The experimental results show that Sonum outperforms ECMP and Hedera in terms of network throughput and flow completion time.

Keywords: packet sampling; long flow detection; load balance; traffic scheduling; heuristic algorithm; data center network

1. Introduction

A data center (DC) is a large-scale internet facility consisting of thousands of interconnected servers for computing and storage, supporting large applications such as web search, social networking and cloud computing. With the increasing demand in recent years, more and more funds have been invested in optimizing the performance of data centers, which has led to a lot of research on improving network performance in industry and academia [1]. One way to improve the performance of a data center network (DCN) is traffic load balancing [2].

The traditional traffic load balancing scheme does not distinguish flows [3], but uses the same scheduling policy for all flows, which makes the load balancing effect extremely poor [4]. From our perspectives, traffic load balancing should be divided into two parts: Traffic information collection and traffic scheduling. Traffic information collection is important for the network management and optimization in data centers, because traffic scheduling requires detailed information on the flows being transmitted [5].

According to previous research [6], traffic in datacenters obeys a long-tail probability distribution. Specifically speaking, nearly 90% of all flows are smaller than 10KB and last under a few milliseconds, while the remaining 10% account for most of the traffic. This means most flows are short flows, meanwhile most bytes are from long flows [7]. Short flows are generated by web search and HTTP requests, which must be transmitted before the deadline of flow completion time (FCT). Long flows are generated by VM migration, data backup, and MapReduce, which applications are sensitive to throughput [8].

Nowadays, data centers usually use the Equal-Cost MultiPath (ECMP) protocol as a basic load balance method. ECMP ensures that all path utilizations are balanced by uniformly distributing the data flows to different transmission paths. However, this mechanism using hashing or polling may result in congested links become more congested. In order to pursue the full use of bandwidth, delay-sensitive short flows tend to be queued after long flows and cannot be effectively forwarded in time. This will increase the FCT of all flows. All in all, An ideal load balancing scheme should design different scheduling schemes for long flows and short flows [9].

The ideal load balance scheme should look like this: The network runs traffic classification to distinguish between long and short flows. After all long flows are found from all flows, it would schedule long flows and short flows for load balancing [10].

Meanwhile, most existing load balancing solutions achieve the goal of minimizing FCT by equalizing link utilization [11]. However, latency and bandwidth can affect each other. In this paper, minimizing network packet loss rate instead of equalizing link utilization is an optimization goal of data center networks. The reason why packet loss rate is used as the target is that it has a great impact on the application of a data center network and will significantly decrease the long tail distribution of FCT.

In conclusion, We conducted research on the following three issues in this paper:

- How to collect network traffic information at a lower cost, especially the long flow and short flow information.
- How to implement dynamic network load balancing for long flows and short flows.
- How much does network performance improve by load balancing based on long flow and short flow information.

Motivated by the above questions, we present a complete solution named Sonum, which contains a synergetic software-defined sampling and detection approach for long flow and an optimal network utilization mechanism for long flow scheduling. Synergetic software-defined sampling and detection approaches could efficiently identify long flows and short flows with lower latency, missed detection rate and communication overhead. Optimal network utilization mechanisms could significantly reduce packet loss and retransmission rate due to switch congestion and link congestion, reduce FCT and improve network throughput. The contributions of this paper are as follows:

- We introduce a software-defined synergetic sampling and detection approach for long flow and analyze its performance in a data center network environment.
- We introduce an optimal network utilization mechanism for long flow scheduling, considering not only link status but also switch status.
- We propose an heuristic long flow scheduling algorithm for Sonum

Experiment results show that Sonum can detect long flows and schedule long flows more efficiently than existing methods. Compared to ECMP and Hedera [12], Sonum performs better in terms of flow completion time and throughput.

The remainder of this paper is organized as follows. Section 2 describes the background of this research and Section 3 reviews the related work. Section 4 details the system design of Sonum from two parts: Synergetic long flow detection approach and long flow scheduling mechanism, and then Section 5 presents the simulation experimental results and discussion. At last, Section 6 concludes the whole paper and points out future work.

2. Background

2.1. Software Defined Networking

The traditional data center network has the characteristics of distributed control, highly coupled control and forwarding, and best-effort forwarding, with the result that complex scheduling strategy is difficult to be deployed. The distributed implementation of scheduling strategy means the network can not be dynamically adjusted according to the current state of the whole network, so more and more people will consider using a software-defined network (SDN) [13], which is a loosely coupled and highly scalable network architecture.

SDN is a new network architecture; its core idea is decoupling of data plane and control plane. Switches are only responsible for simple data forwarding. Decision-making functions are all concentrated in the controller. As the core of an SDN, the controller can control the entire network and communicates with switches through OpenFlow message. SDN architecture can develop a flexible network forwarding strategy for complex and changeable business scenarios, which provides a new idea for the data center traffic load balancing problem.

2.2. Data Center Network

The Fat-Tree architecture is currently the most popular network architecture in a data center. Figure 1 shows a typical Fat-Tree architecture. Sixteen groups of servers (S1–S16) are linked to the edge switches (E1–E8), which is the bottom layer of Fat-Tree. The second layer is composed of eight aggregation switches (A1–A8), and the top layer is four core switches (C1–C4).

An edge switch and a set of servers connected to it are called a pod. Therefore, the topology in Figure 1 contains eight pods. According to whether the traffic crosses the pod, the traffic in the data center can be divided into two types: Inter-pod flows and internal pod flows. The sender and receiver of an inter-pod flow belong to two different pods. Internal pod flow is the flow of communication between two servers under the same edge switch.

Generally, the traffic collection function is deployed on the edge switches. Their independent work has made entire network traffic collection very inefficient. However, we have found that an inter-pod flow is forwarded by at least two edge switches and two aggregation switches. This means that packets belonging to one flow have at least four chances to be sampled ideally. So if we can allow each switch to run a single layer of software sampling, and the switches can exchange their own sampling information, then the overall network sampling effect should be improved [14].

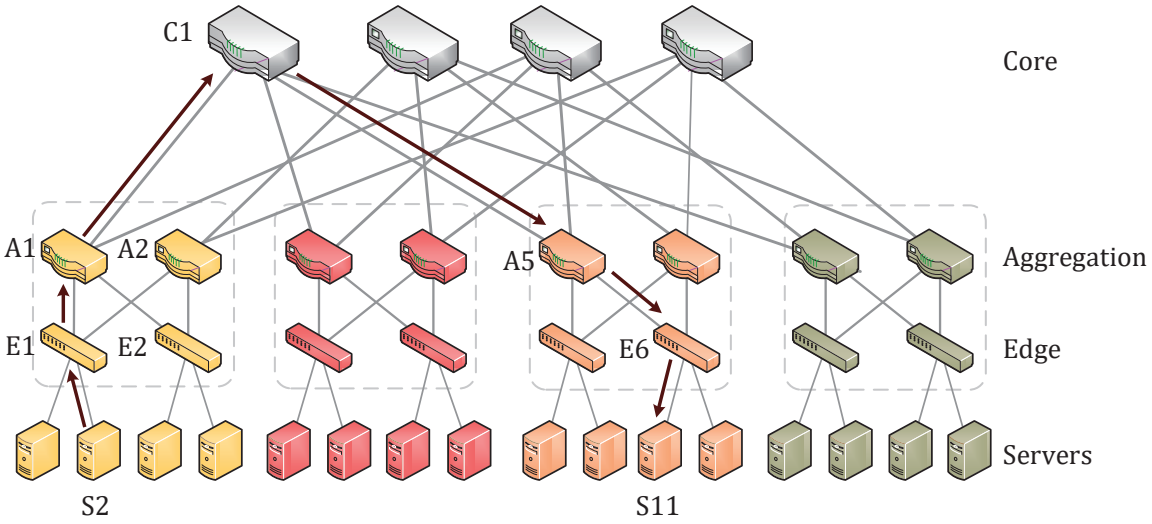

Figure 1. Data center network with multi-tier Clos topology (Fat-Tree).

In a data center, there is a large number of optional paths from a source server to a destination server. These optional paths generally have the same cost metric in the routing table of the switch. Therefore, switches typically use ECMP mechanisms (IP 5-tuple hashes) to spread traffic across different optional paths.

2.3. Load Balancing

The key of data center network traffic load balancing is to study how to distribute traffic to these equivalent links as evenly as possible. Most of the research work used link utilization as the optimization goal. However, this goal is not appropriate in data center networks. There are four reasons:

1. Link utilization is a statistical average value, which requires continuity of network traffic on a time scale. However, data center network traffic characteristics do not meet such needs. Devices such as servers and switches in the data center network are usually only distributed in a small physical area, so the transmission delay is lower than the Internet. Therefore, the switch buffer pool size is typically much larger than the number of packets being transmitted on the link.
2. In data center networks, many-to-one or many-to-many communication modes are widely found in the parallel computing systems (MapReduce, Dryad and CIEL) and distributed file storage systems (GFS and HDFS). In order to adapt to these special communication modes, the data center transmission protocol has also changed compared to the Internet backbone. Even so, there are still TCP Incast and TCP Outcast problems [15]. TCP Incast is an uncontrolled TCP behavior that causes considerable link underutilization in many-to-one communication paradigms [16]. TCP outcast is that when traffic that consists of connections of many incoming flows and few flows at two ports switch (or router) and compete to one common output port, throughput the few flows is affected implicitly [17]. The main reason for these two problems is competing port resources. These will not happen on the Internet backbone.
3. The data center network contains a large number of short burst flows. The related measurement work [18] used Micro-Burst to describe these short burst flows. The results showed that most of the bursts have a duration of less than 10 s. Other measurements [19] showed that 90% of burst durations do not exceed 2 s. These traffic bursts are strong, difficult to predict, and have a short duration. Internet backbone network traffic engineering does not consider this situation. The traffic in the backbone network is generally regular and relatively stable. This is the main difference between a backbone network and a data center network. Many previous studies have used the backbone network approach to the data center. What we want to say is that the

characteristics of these two networks are different, so different research ideas need to be used to solve load balance problems.

4. The communication between data center servers is widely used in the TCP protocol, accounting for 95% of the total traffic [20]. TCP is designed for low-bandwidth, low-latency WANs, and is not suitable for high-bandwidth, low-latency data center networks. Even though there is a variety of TCP modified transport protocols for data center networks, congestion is inevitable in these protocols, which do not exist on the Internet backbone.

Therefore, we propose minimizing network packet loss rate instead of balancing link utilization optimization factor of data center network traffic load balancing.

The loss of packets is a systemic problem of the network [21]. There are five main reasons for packet loss in the data center network:

1. Unreachable Route

   This paper does not consider packet loss due to an unreachable route.

2. Equipment Failure

   This paper does not consider packet loss due to equipment failure.

3. Poor Link Quality

   Packet loss caused by poor link quality can be solved by improving link quality. Traditional data center internal networks often use lossy links, but many companies have tried to deploy lossless network in data centers [22].

4. Link Congestion

   Link congestion refers to the random packet loss caused by the data that is not forwarded in time when the amount of data to be forwarded on one link exceeds the link capacity. Both academia and industry are paying close attention to the congestion of traditional lossy data center networks and designing a series of congestion control mechanisms. The well-known DCTCP protocol can effectively control the switch queue length when network congestion is not serious [20], but DCTCP is still difficult to avoid overflow of the switch buffer when a large number of concurrent links occur. Therefore, these mechanisms are also difficult to avoid congestion and packet loss.

5. Switch Congestion (Switch Buffer Overflow)

   Whether the network uses electrical signal transmission or optical signal transmission, when the data packet enters the switch, data basically needs to be buffered. Usually, the optical signal also needs to be converted into an electrical signal for processing, and it is difficult to directly process the optical signal. The electrical signal processing speed of the switch is limited by the processing capability of the device. As shown in Figure 2, when forwarding the data packet, the packet needs to be parsed, repackaged and forwarded. These operations need to be performed in the memory. Moreover, caching the data packets requires a large amount of storage space. When different types of flows compete for the switch cache, the long flow will occupy most of the switch cache queues space, making the latency performance of the short flow become worse.

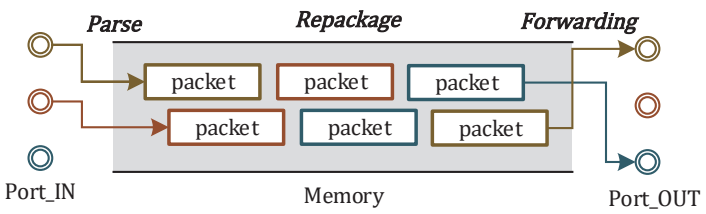

Figure 2. Shared memory switch mode in data center network.

With the parallel processing, shared buffer space, and buffer management scheduling technology, such large memory is often not needed. Even so, data center switches are designed with large memory features.

In conclusion, packet loss is the most direct cause of the increase of FCT, and the main reasons for packet loss are link congestion and switch congestion. It is not appropriate to only minimize the maximum link utilization for the entire network, and we should consider both link resources and switch resources. Link resources can be represented by link utilization. Switch resources can be represented by switch load. This paper tries to use link remaining bandwidth and switch idle memory size to represent link resource and switch resource. Further, we refer to link resources and switch resources collectively as network resources. In this paper, data center load balancing can be described as the matching problem between traffic and network resources, that is, the overall load balancing of the network is realized on the basis of minimizing the network packet loss rate. It also naturally achieves the traditional goal of minimizing the maximum link utilization in this case.

3. Related Work

We classify the prior work on long flow load balancing into two parts: Traffic information collection and traffic scheduling.

For traffic information collection, the simplest separation between long and short flows has to pay a high price in practice, due to the expansion of datacenter server scale and the acceleration of network rate. At present, there are three categories methods of long flow detection. As shown in Table 1, They are active measurement, passive statistics and sampling.

1. Active measurement

   Active measurement method is usually realized by Openflow protocol [23], such as Hierarchiical Statistics Pulling [12]. The SDN controller sends Read-State messages periodically to the switches, then receives feedback containing the packets, bytes and duration for each flow. If the controller sets a long flow detection threshold and the received result of a flow exceeds the threshold, the flow is diagnosed as long flow. Nevertheless, collecting flow information on each switch brings huge communication cost to the controller.
2. Passive statistics

   The passive statistics method requires changing switch hardware, by adding a statistical module to each switch. Like DevoFlow [24], The function of statical module is to count the information on each flow. If the statistics of a certain flow exceeds the threshold of the long flow detection, the notification to controller is triggered. This method reduces the communication cost between controller and switches.
3. Sampling

   The sampling method is a universal approach to collect traffic information. However, for long flow detection, the existing solution often does not meet the requirements of long flow detection accuracy and suffers from feasibility issues [25]. FlowMon fisrt captured the suspicious long flow through coarse-grained sampling method optimized the TCAM resource allocation [26].

For traffic scheduling, there are two main methods for solving datacenter network long flow load balancing, namely, slice spraying and active scheduling.

Table 1. Comparison of current long flow detectiong methods.

Method	Accuracy	Cost	Complexity	Hardware Changes
Active measurement	High	High	Easy	No
Passive statistics	High	Low	Difficult	Yes
Sampling	Low	Mid	Easy	Yes

Slice the spraying method forwards flow at the packet level, making the load on each link balanced, e.g., TinyFlow [27]. However, slice spraying can cause packets thar are out of order, which affects the actual performance of the network. FlowBender [28] advocated Load balances distributively at the granularity of flows instead of packets, avoiding excessive packet reordering [29].

For active scheduling, previous researchers have also given many different solutions. Hedera [12] was a centralized dynamic traffic scheduling algorithm, using a fixed polling cycle to sample network traffic, which is the earliest research paper based on the SDN architecture. By default, Hedera uses ECMP to forward traffic. After the switch discovers a long flow in the network, it sends the long flow information to the centralized scheduler. The scheduler will perform unified scheduling on all the long flows, and deliver the scheduling results to the switches in the form of OpenFlow flow entries. The switches forward the long flows indicated by the flow entries in the flow table. Otherwise, the switch forwards according to the ECMP mechanism. Hedera's centralized scheduling algorithm is very effective in theory, however, it does not completely solve the problem of traffic balancing in practice. The main reason for this phenomenon is that Hedera assumes competition between long flows leads to network congestion. In fact, short flow can compete due to high instantaneous rates, resulting in network congestion. Moreover, Hedera's long flow detection mechanism will also increase the network burden. DiffFlow was an improved scheme based on the Hedera algorithm. DiffFlow [30] also uses packet sampling to perceive the existence of long flows in the network, but does not explore the sampling performance in detail. Other works include MircoTE [31], OpenSample [32], Sample&Pick [33], etc. These solutions focused on data center network load balancing, and did not have a good answer to how to obtain network long flow information.

Freeway [34] proposed a dynamic path partitioning algorithm to adjust dynamically with varying traffic load the number of low latency and high throughput paths. While mice flows are transmitted over low latency paths using a simple equal cost multiple path (ECMP) scheduling, Freeway loaded balances elephant flows on different high-throughput paths. Freeway is actually a resource reservation solution. Similar to Freeway, PIAS [35] minimized the FCT by mimicking shortest job first (SJF) on the premise that flow size is not known a priori, then leveraged multiple priority queues available in existing commodity switches to implement a multiple level feedback queue. Based on PIAS [36], LinkGame applied two-sided matching decision in game theory to solving traffic scheduling problem in data center network. Considering the preference ordering, path-flow matching problem was formulated as a multiobjective optimization problem with the target to ensure the stability and satisfaction from the matching scheme.

Some works had also considered the impact of switches on load balancing, such as SAB [37] and Fincher [38]. SAB utilized the characteristics of that the switch buffer pool, usually much larger than the round-trip delay bandwidth product in the data center network. SAB set the congestion window value of the flow by assigning the switch cache. This scheme reduced the short flow completion time and effectively solved the problems of TCP Incast and TCP Outcast. However, SAB sacrificed the performance of long flow to make the short flow completion time shorter, so it is not suitable for long flow. Fincher proposed that the mapping between data center traffic and switch memory is a many-to-one stable match problem and proved this problem is an NP-hard problem. Fincher considered the impact of switch resources on flow completion time, and did not consider long flow scheduling through multiple paths.

There are also some studies that transfer long flow detection and scheduling from the switch to the server. Mahout [39] obviously had higher sampling accuracy but this was not compatible with existing networks. AuTO [40] developed a two-level deep reinforcement learning system, mimicking the Peripheral & Central Nervous System, to solve the scalability problem. Peripheral System resided on end-hosts, collect flow information, and make traffic optimization decisions locally with minimal delay for short flows. Central System aggregated and processed global traffic information and made individual traffic optimization decisions for long flows. However, the latency of deep reinforcement learning systems is the major obstacle to traffic optimization at the scale of data centers.

4. System Design and Analysis

In this section, we will detail and analysis Sonum from two parts: The synergetic software-defined sampling and detection approach for long flow and the optimal network utilization mechanism for long flow scheduling.

Sonum has four modules: Flow Collector, Flow Counter, Long Flow Detection and Long Flow Scheduling. The function of the first three modules is discovering long flow. The function of the last module is re-routing the discovered long flow.

As shown in Figure 3, A packet of a certain flow enters the openflow switch and performs normal forwarding according to flow table. Based on the sampling rate v_s, the Flow Collector Module determines whether to send the protocol header of the packet to the Flow Counter Module for processing.

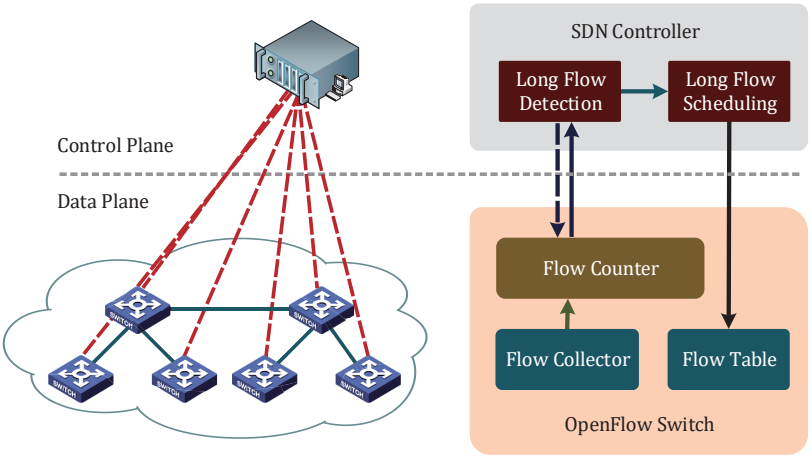

Figure 3. The system model of Sonum.

In step 1, The Flow Counter matches the header of the protocol submitted by the Flow Collector with the established entry. If there is a match and the counter does not overflow, the counter increments. If there is no match, a new count entry is created according to the protocol header data. As shown in Figure 4, in order to facilitate lookup and counting, each flow is identified by 5-tuple: Source/destination IP, source/destination port number and transport protocol. Flow Counter Module perform a hash calculation on the 5-tuple of each flow, then record the flow information and count result plus 1.

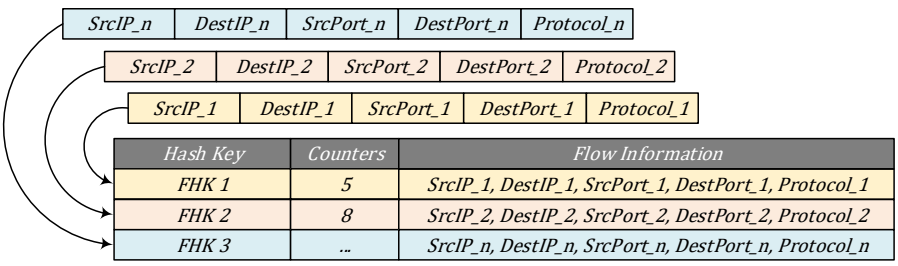

Figure 4. Count operation of flows.

In step 2, the controller broadcasts the detection parameters according to the network status, including the sampling rate v_s and the long flow threshold T_l. OpenFlow switch sends all the flow information whose counter value is greater than T_l to the Long Flow Detection Module of the controller.

In step 3, the controller caches the flow information and determines whether the flow is a potential long flow or not. If yes, the Long Flow Detection Module sends the flow information to the Long

Flow Scheduling Module. Long Flow Scheduling Module generates flow tables according to optimal network resource utilization mechanism and send flow tables to all switches.

Steps 1–3 can detect and schedule long flows. Specifically, they can be realized through interaction information between different switches.

4.1. Synergetic Software-defined Sampling and Detection Approach

Synergetic software-defined sampling and detection is the foundation of Sonum. The emergence of a Software-Defined Networks makes software traffic collection possible. Compared to hardware traffic collection, software solutions are more flexible to deploy and can dynamically adjust sampling rate as needed. The disadvantage of software solutions is that performance is often not good. On the basis of existing network topology and transmission mode, through collaborative optimization, using a lower sampling rate to achieve a better sampling effect is the goal of software traffic information collection. The responsibility for synergetic software-defined sampling and detection is to effectively detect long flow using a lower sampling rate and a lower communication cost between controller and switch. Therefore, we designed three mechanisms to achive this goal: Synergetic Sampling, Self-Adaption Sampling and Lifecycle Management.

The problem of long flow detection is as follows: Given a threshold parameter T_l and a traffic information $F = F_1, F_2, \ldots, F_n$, a long flow is a flow F_i includes more than T_l percent of packets since the beginning of the transmission.

4.1.1. Synergetic Sampling

The sampling probability of the switch that a flow passes through can be expressed as:

$$P(S_i) = e^{\frac{-\Gamma}{S_i}}, S_i \geq 0 \tag{1}$$

where Γ is the average of the statistics of each switch, S_i is the sampling statistics of each switch for this flow. S_i is related to sampling rate v and all flow number m. For any switch, if preset flow statistics threshold is γ. the probability that the statistical result is greater than the threshold is The sampling probability of the switch that a flow passes through can be expressed as:

$$P[S_i \leq \gamma] = 1 - e^{\frac{-\Gamma}{S_i}} \tag{2}$$

The joint probability that all switch statistics are less than the threshold is

$$P[S_{1,2,\ldots,n} < \gamma] = (1 - e^{\frac{-\Gamma}{S_i}})^n \tag{3}$$

where n is the number of switches that deploy synergetic software-defined sampling approach. We are concerned about whether the long flow can be detected, so the probability that at least one switch detected the long flow is

$$P[S \geq \gamma] = 1 - (1 - e^{\frac{-\Gamma}{S_i}})^n \tag{4}$$

If the network has already been running ECMP, we can use the path as the basic unit instead of the switch to consider the relevant probability.

4.1.2. Self-Adaption Sampling

Almost all of the conventional sampling schemes rely on fixed period (e.g., NetFlow) or a fixed percentage (e.g., Hedera). The advantages of these two fixed samplings are reducing the sampling operation overhead of the switch and facilitating the deployment of the sampling strategy. However, the disadvantage is missing important data packets.

If the purpose of sampling is long flow detection, a self-adaption sampling approach can adjust the sampling rate dynamically according to the packet arrival rate and the length of the sampled queue. It is generally believed that the abnormal increase of network traffic is usually caused by long flow transmission. Therefore, the increase in packet arrival rate implies that a potentially long flow is generated and being transmitted. For ease of deployment, Sonum cannot take up too much storage resources. The queue length indicates the number of statistics in a table. As the sampling progresses, queue length also fluctuates. The increase in the average queue length means that the number of statistics has been added to the already established list. The decrease in the average queue length means that the new sample item is established.

Taking into account the above two parameters, we have established a formula for the relationship between the two parameters and the sampling rate. First, the synergetic software-defined sampling approach windowed historical sampling rate data.

$$R = \{v_1, v_2, ..., v_p\} \tag{5}$$

where p is the window size, and v_i is the sampling rate when $t = i$. Then this approach calculates the statistical average of the packet arrival rate and the average queue length in the window period, which are $\bar{r}$ and $\bar{l}$. According to the change of packet arrival rate and queue length at the latest sampling time, this approach determines the sampling rate for the next cycle, as shown in the Equation (6).

$$R_{p+1} = \frac{R_p}{2}(\frac{r_p}{\bar{r}} + \frac{l_p}{\bar{l}}) \tag{6}$$

where R_p is the sampling rate of the previous sampling period. As long flow detection needs to meet the demands of the traffic characteristics, the sampling rate should not be too slow. To make the sampling rate adjustment smoother, this approach uses the rate adjustment by means of an Exponentially Weighted Moving Average (EWMA) [41] , as shown in Equation (7). EWMA is a commonly used method of sequential data processing. By weighting historical data and recent observations, EWMA makes the rate adjustment smoother, avoiding the situation that the sampling rate is repeatedly adjusted due to the rapid change of the packet rate.

$$R_{p+1}^{EWMA} = \lambda \cdot R_{p+1} + (1 - \lambda) \cdot R_p, 0 < \lambda < 1 \tag{7}$$

where λ is the smoothing parameter. The bigger λ is, the sampling adjustment function is more concerned with the latest sampling data. The complete Rate Adjustment Algorithm is described in Algorithm 1.

Algorithm 1 Rate adjustment algorithm (RAA)

Require: Historical sampling rate data R, Smoothing parameter λ
Ensure: Next sampling rate data R_{p+1}
1: Calculate R_{p+1} according to Equation (6);
2: **if** $R_{p+1} \neq R_p$ **then**
3: Calculate R_{p+1}^{EWMA} according to Equation (7);
4: $R_{p+1} = R_{p+1}^{EWMA}$;
5: **else**
6: $R_{p+1} = R_p$;
7: **end if**
8: Return R_{p+1};

4.1.3. Lifecycle Management

In order to implement Sonum, we need to optimize the sampled information in order to reduce the need for storage resources. Deleting old entries and reserving storage space for new flow is an

effective way. With reference to the relevant provisions of the OpenFlow protocol timeout release, we perform probabilistic selective deletion of entries whose statistics are much lower than the sampling threshold, and delete flows with longer dwell times.

4.2. Optimal Network Utilization Mechanism

Sonum attempts to analyze the load balancing problem in the datacenter network, define the purpose of optimizing network utilization and give the feasible solution.

4.2.1. Long Flow Scheduling and Network Utilization Analysis

Load balance based on ECMP can only be achieved if the network is symmetric, because ECMP is stateless. Under the influence of the randomness of the hash algorithm and flow size, the distribution of network traffic at a certain moment in an entire network is not uniform. ECMP cannot cope with network topology asymmetry caused by switches and link failure. At the same time, Uniform traffic distribution tends to result in insufficient link bandwidth, which is a hot spot as shown in Figure 5. As an example, assume there are n long flows with a peak bandwidth requirement of d in the network, and distribute them in m links with capacity c. If $nd > mc$, there will be an overall bandwidth shortage. If $nd < mc$, there will also be a link congestion. All in all, ECMP protocol is less flexible.

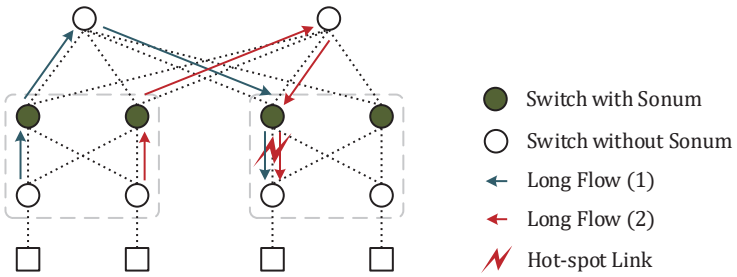

Figure 5. Host-spot Link and Deployment of Sonum.

The performance of the entire network can be measured by two main metrics, the FTC and throughput. The goal is to minimize FCT for shot flows, while maintaining an acceptable throughput for long flows. The FCT is described as the interval between the start time stamp t_s when the first packet of a flow leaves the source server and the end time stamp t_e when the last packet of the flow arrives at the destination server [30]. The ideal FCT is the best achievable where only the service times of the intermediate nodes and the transmission time of the packets are considered. Because it is not easy to get the FCT of the network, previous studies have indirectly measured the flow completion time through bandwidth, and the more balancing the bandwidth utilization, the shorter the FCT.

As described in Section 2, minimizing network packet loss rate instead of balancing link utilization is more suitable for data center network traffic load balancing. Packet loss is the most direct cause of the increase of FCT, and the main reason for the packet loss are link congestion and switch congestion. We should consider both link resources and switch resources to reduce packet loss rate. Link resources can be represented by link remaining bandwidth. Switch resources can be represented by switch idle memory. Data center load balancing can be described as the matching problem between traffic and network resources, that is, the overall load balancing of the network is realized on the basis of minimizing the network packet loss rate.

It is worth noting that the utilization of network resources for each available path should be described as the maximum link or switch utilization on the path rather than the sum of link or switch utilization [42].

4.2.2. Optimal Network Utilization Model

As mentioned above, there are three objects involved when performing long flow scheduling in a data center. They are long flows, switches and links. Long flows are flows whose packet number is greater than threshold. Switches is the set of entire network switches, including core switches, aggregation switches and edge switches. Links is the set of all links that connect switch to server and connect switch to switch. Some denotations and symbols used in optimal network utilization model and algorithm are described in Table 2.

Table 2. Denotations used in model and algorithm.

Denotations	Description
F	The set of all long flows
P	The set of all paths in entire network
S	The set of all switches
L	The set of all links
l	The number of network layers
n	The number of all servers
$P(s,d)$	The set of all paths from server s to server d
$p(s,d)_i$	The ith path in $P_{(s,d)}, i = 1, 2\ldots, \lvert P\rvert$
$l(s,d)_{ij}$	The jth section of ith path in $P_{(s,d)}, i = 1, 2\ldots, \lvert P\rvert$, and $j = 1, 2, 3, 4$ if $l = 3$
$s(s,d)_{ik}$	The kth switch of ith path in $P_{(s,d)}, i = 1, 2\ldots, \lvert P\rvert$, and $k = 1, 2, 3, 4, 5$ if $l = 3$
$lc(s,d)_{ij}$	The link capacity of jth section of ith path in $P_{(s,d)}$
$ll(s,d)_{ij}$	The link load of jth ection of ith path in $P_{(s,d)}$
$mc(s,d)_{ik}$	The memory capacity of kth switch of ith path in $P_{(s,d)}$
$mf(s,d)_{ik}$	The memory footprint space of kth switch of ith path in $P_{(s,d)}$
f_m	The mth flow in $F, n = 1, 2..., \lvert F\rvert$
r_m	The rate of long flow f_m
M	The matching between paths and flows

Then we can define the Link Resource Utilization (LRU) of jth section in path $p(s,d)_i$:

$$LRU_{p(s,d)_i} = \frac{ll(s,d)_{ij}}{lc(s,d)_{ij}}, j = 1, ..., 2^{l-1} \tag{8}$$

the Switch Resource Utilization (SRU) of the kth switch in path $p(s,d)_i$:

$$SRU_{p(s,d)_i} = \frac{mf(s,d)_{ik}}{mf(s,d)_{ik}}, k = 1, ..., 2^{l-1} - 1 \tag{9}$$

So, the Network Resource Utilization(NRU) of path $p(s,d)_i$:

$$NRU_{p(s,d)_i} = \max\{LRU_{p(s,d)_i}, SRU_{p(s,d)_i}\} \tag{10}$$

$NRU_{p(s,d)_i}$ contains $j + k$ parts: j LRUs and k SRUs.

Therefore, the long flow scheduling problem of the data center network can be described as:

Given the network paths set of $P = \{P_{s_1,d_1}, ..., P_{s_n,d_{n-1}}\}$ and long flow set $F = \{f(s,d)_1, f(s,d)_2, ...\}$, to find a matching $M_{F,P} = \{f(s,d)_m, p(s,d)_i | f(s,d)_m \in F, p(s,d)_i \in P(s,d)\}$ with minimum entire NRU.

Obviously, NRU can also be expressed as

$$NRU \Longleftrightarrow \sum_s \sum_d \sum_i NRU_{p(s,d)_i} \tag{11}$$

In summary, long flow scheduling can be described as:

$$min \quad \sum_{s}\sum_{d}\sum_{i} NRU_{p(s,d)_i} \tag{12}$$

$$s.t. \quad (lc(s,d)_{ij} - ll(s,d)_{ij}) \geq \sum_{f_m \in M(p(s,d)_i)} r_m \tag{13}$$

$$(mc(s,d)_{ik} - mf(s,d)_{ik}) \geq \sum_{f_m \in M(p(s,d)_i)} r_m \tag{14}$$

$$E(f_m, M(f_m)) = 0 \tag{15}$$

$$|M(f_m)| \geq 1 \tag{16}$$

$$1 \leq j \leq 2^{l-1}, j \in Z \tag{17}$$

$$1 \leq k \leq 2^{l-1} - 1, k \in Z \tag{18}$$

$$1 \leq s, d \leq 2^n, s, d \in Z \tag{19}$$

The goal of long flow scheduling is to minimize the overall network resource utilization, including switch resource and link resource. The constraint (13) is to ensure that all links do not saturate. The constraint (14) is to ensure that all switches do not experience memory overflow. The constraints (15)–(16) are to ensure that each long flow should be mapped to at least one path. This mapping problem has been proven to be an NP-Hard problem.

4.2.3. Heuristic Somun algorithm

The goal of long flow scheduling is to find the best candidate from all optional paths for each long flow [43]. The heuristic long flow scheduling algorithm HSA is described in Algorithm 2. The inputs of HSA are long flow set F and path set P, and the output of HSA is the mapping solution set M. Firstly, HSA initializes the mapping set M and polls the set F. Secondly, When the set F is not empty, it means that a long flow is detected and the long flow needs to be scheduled. HSA calculates the SRU for each path based on the source and destination addresses of the long flow. If there is a suitable path that satisfies the constraints and is available for allocation, HSA deletes this long flow from F, and records the matching relationship between the long flow and the path in M. Finally, HSA updates the path information, outputs the mapping relationship, and generates a flow table for delivery.

HSA's time complexity is O(n). The speed of HSA is related to the size of long flow set F and path set P. The processing of the HSA is extremely simple, which means that the HSA does not need to consume too much computing resources of the controller, and can quickly generate routing policies.

5. Performance Evaluation

In this section, we first analyzed realistic performance of the Synergetic software-defined sampling spproach. Then, we analyzed the sampling and detection performance of Sonum from the aspects of missed detection ratio and storage requirements. Finally, we evaluated Sonum compared with ECMP and Hedera.

We have implemented the functional modules of Sonum that is described above using ONOS, Mininet in Ubuntu 16.04. As shown in Figure 1, the simulated topology has four core switches, eight aggregation switches, eight edge switches and 16 servers. ONOS is a lightweight controller that have the enough processing power of acquisition, so we only deployed Sonum Sampling on edge switches A1 and A5 to detect long flow information in above topology. The detailed simulation parameters can be found in Table 3.

Algorithm 2 Heuristic Scheduling Algorithm (HSA)

Require: Long flow set F, Path set P
Ensure: The mapping solution M between paths and long flows F, P

```
M = ∅;
while F ≠ ∅ do
  for i = 0 → (|F| − 1) do
    Obtain the source and destination addresss s, d of f_i ∈ F;
    Calculate the SRU_{p(s,d)_i} according to Equation (10);
    Sort increasingly SRU_{p(s,d)_i} according to Equation (13) to 19 and save them to an array candidate;
    if candidate ≠ ∅ then
      Generate optional path p(f_i) according to min(SRU_{p(s,d)_i});
      F = F\f_i and M = M ⋃ (f_i, p(f_i));
      Update link load ll(s,d)_ij and memory footprint space mf(s,d)_ik according to Equation (13) and Equation (14);
    else
      Return mapping failed;
    end if
  end for
  Generate route policy according to M;
end while
```

Table 3. Simulation Parameters.

Parameters	Value
Core Switches Number	4
Aggregation Switches Number	8
Edge Switches Number	8
Server Number	16
Max Packet Size	1500 Bytes
Switches Interface Rate	1 Gbps
Short Flow Size	1 KB–100 KB
Long Flow Size	100 KB–2000 MB
Long Flow Size	100 KB–2000 MB
Short Flow Number:Long Flow Number	1:9
Simulation Time	20 min
Initial Sampling Rate	1000:1, 10,000:1, 100,000:1

5.1. Performance of Synergetic Software-defined Sampling Approach

The long flow detection result is shown in Figure 6. Figure 6a is the result of switch A1 and Figure 6b is switch A5. Although both switches undetect some long flows, the controller can still further reduce the missed detection ratio based on switches sampling information and the long flow detection results. The circled flow in Figure 6c is the flow that have been undetectable. In traditional detection schemes, the cost of these streams is detected to increase the sampling rate. In Sonum, they are discovered because the exchange of information between switches, rather than the increase in the sampling rate. Sonum reduces the consumption of hardware resources.

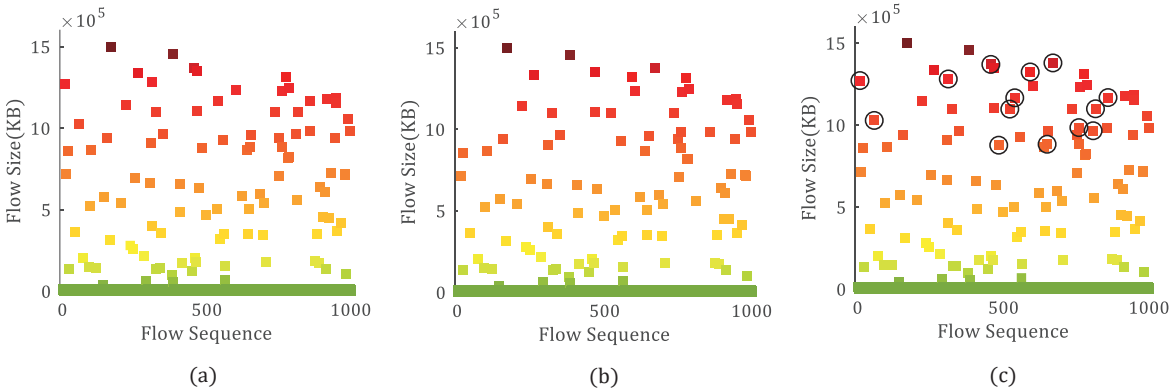

Figure 6. Long flow detection result of top 1000 flows. (**a**) Switch-A1. (**b**) Switch-A5. (**c**) Controller.

We compare the missed detection ratio of Sonum with FlowMon [26] in Figure 7. FlowMon is a sampling method based on SDN. The results show that both have achieved better sampling results at higher sampling rate. The missed detection rate of Sonum reduced by 2.3%–5.1% compared with FlowMon.

As shown in Figures 8 and 9, we compare the susceptibility of three different solutions to long flow detection: Regular sampling, Sonum (no RAA) and Sonum. We use the average number of packets processed until determining long flow in the network as the measure result of sensitivity. Sonum significantly reduces the waiting time required to detect long flow.

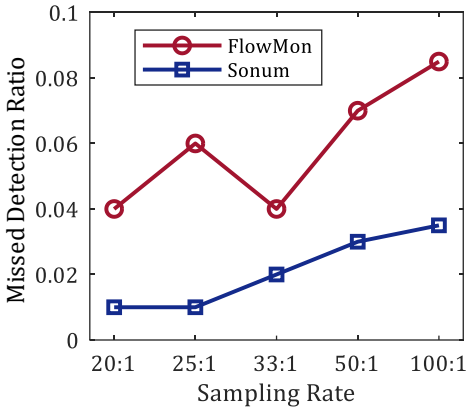

Figure 7. Comparison of Sonum with FlowMon.

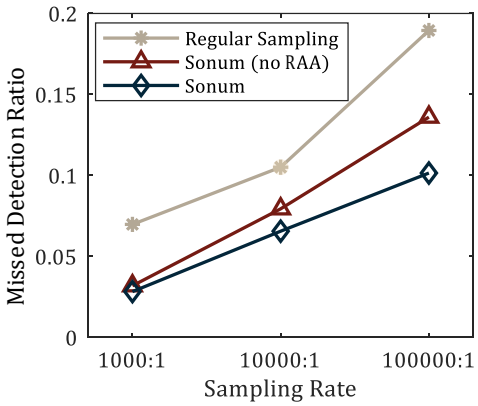

Figure 8. Comparison result of missed detection ratio.

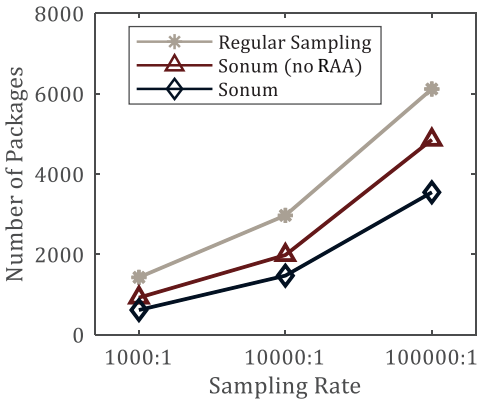

Figure 9. Comparison result of number of packages.

Figure 10 shows the Sonum sampling performance when $\lambda = 1$ and $\lambda = 0.6$. $\lambda = 1$ means there is no smoothing. After EWMA smoothing ($\lambda = 0.6$), the sampling rate adjustment becomes more stable. This operation work better in high-speed situations where the flow rate change frequently, as shown in Figure 10c.

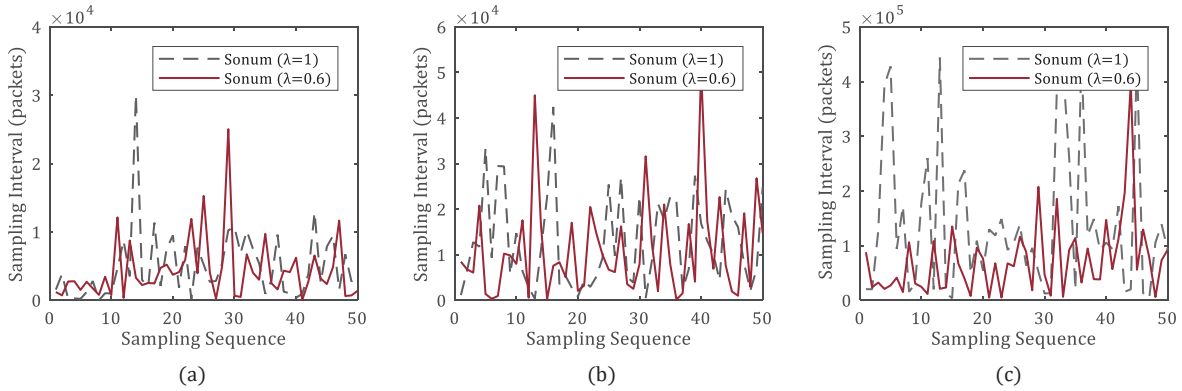

Figure 10. Statistic results of sampling interval when different λ. (**a**) sampling rate = 1000:1. (**b**) sampling rate = 10,000:1. (**c**) sampling rate = 100,000:1.

Figure 11 shows the storage requirements of Sonum. The primary factor related to the queue length is the sampling rate. The larger the sampling rate, the longer the queue length. This is because the large sampling rate causes some short flows to be sampled and occupy the buffer queue for a long time. In addition, the sampling rate is larger, the sampling accuracy is higher, and the number of long flows in the storage queue is also larger, which is another reason. The controller has a longer cache queue than the switches because the controller needs to process the long flow information reported from switches. The storage overhead that the controller needs to maintain is greater. Our experiment only counts the deployment of Sonum between switch A1 and switch A5. If all the switches in the data center network need to deploy Sonum, the length of storage queue that the controller needs to maintain will be extremely longer. However, the length of queue is not longer than the number of all long flows in the entire network.

Figure 12 shows the queue length fluctuation of switch A1. Due to the addition of the life cycle management mechanism, the queue length change can be stabilized within an acceptable range.

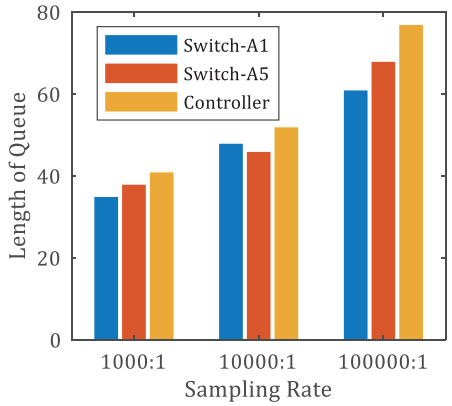

Figure 11. Results of queue length.

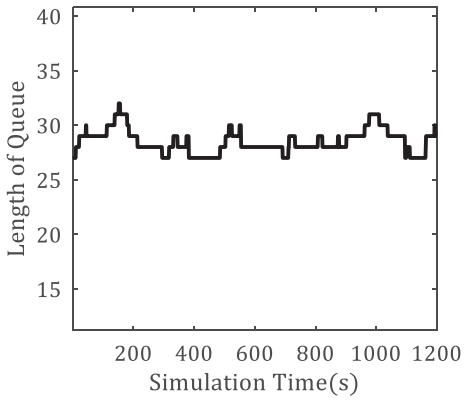

Figure 12. Queue length fluctuation (sampling rate = 1000:1).

5.2. Performance of Optimal Network Utilization Mechanism

We have simulated the traffic processing of the shared memory switch through the setting of the single queue of the switch in Mininet, so as to be added as the switch resource to the verification of this mechanism. Link resources are represented by bandwidth. We have implemented Sonum on ONOS controller, which is connected to the Mininet. It should be noted that the following experiments were performed at a sampling rate of 10,000:1.

Similar to previous research [12], Sonum generates traffic using a probabilistic model, stag(p, q) model, stride(i) model and random model. Sonum adopts ECMP for short flows and HSA for long flows. Both traffic models are implemented by calling iperf in Mininet. We have evaluated and compared the performance of Sonum, ECMP and Hedera [12] on standardized average bisection bandwidth and FCT.

Figure 13 shows the overall standardized average bisection bandwidth performance of ECMP, Hedera and Sonum. In most of traffic models. Sonum has the best performance. Under the stag(1, 0) model, all three schemes have achieved almost the same results. The reason is that this traffic model does not generate traffic across core and aggregation switches. Almost all of the three schemes use the ECMP method for traffic scheduling. In this model, all three schemes are equivalent to ECMP. In other words, the performance of all scheduling schemes is related to traffic models.

In order to test the impact of different schemes on the long flow FCT, we set all the long flow sizes in the network to 100 MB, and all the short flows are set to 50 KB. Figure 14 shows the long flow FCT of three schemes. Sonum has achieved the best performance in all traffic models.

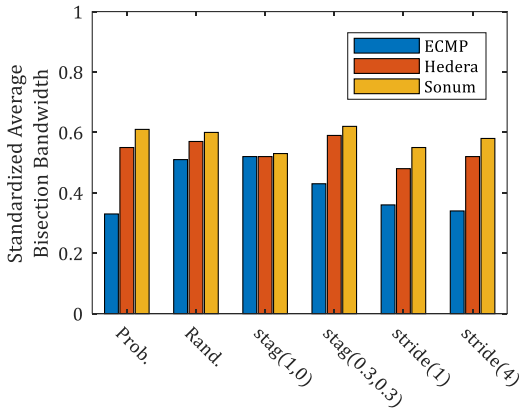

Figure 13. Result of average bisection bandwidth of different load balancing schemes under different traffic models.

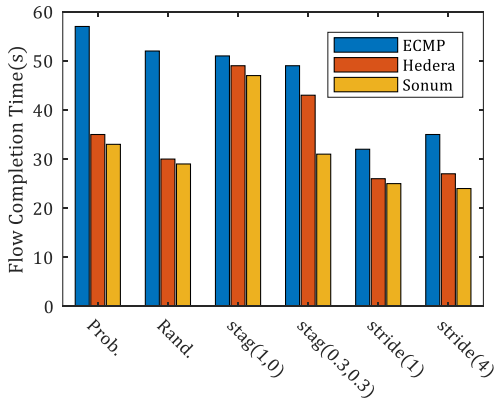

Figure 14. Result of long flow FCT of different load balancing schemes under different traffic models.

Figure 15 shows the improvement performance of standardized average bisection bandwidth of Sonum. Sonum increases bandwidth utilization by 18.3–36.9% compared to ECMP, and 2.1–13.5% compared with Hedera except stag(1, 0) model.

Figure 16 shows the improvement performance of long flow FCT of Sonum. Sonum decreases FCT by 10.3–56.1% than ECMP, and 2.4–40.2% than Hedera except stag(1, 0) model.

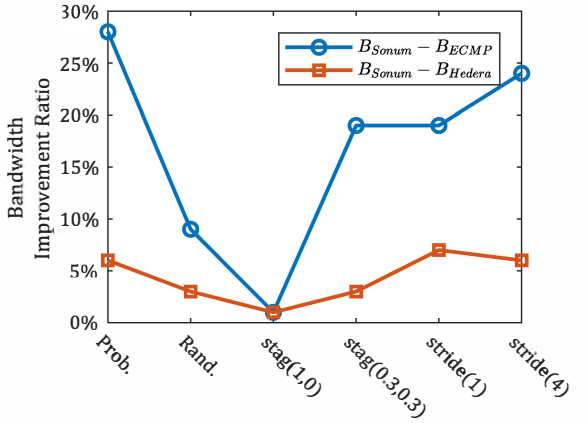

Figure 15. Improvement Performance of standardized average bisection bandwidth of Sonum compared with ECMP and Hedera.

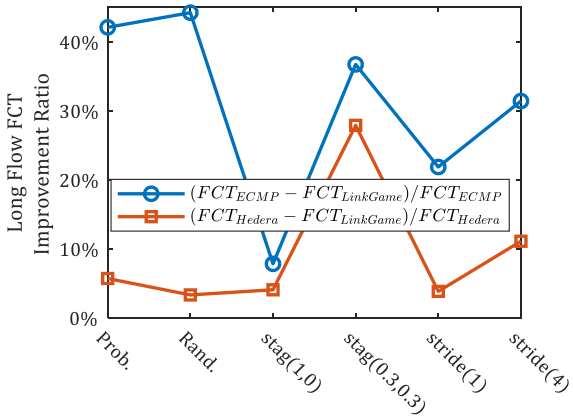

Figure 16. Improvement Performance of long flow FCT of Sonum compared to ECMP and Hedera.

Figure 17 shows the packet loss rate results of three schemes. In order to be convincing, we set the link loss rate in all schemes to 1%, and used iperf to record the network packet loss rate. Simulation results show that Sonum has the lowest packet loss rate. ECMP lacks congestion awareness, which may increase network congestion and further lead to network packet loss. Also, ECMP cannot avoid packet loss due to link congestion and switch congestion. Both Hedera and Sonum schedule on the long flow, which can avoid the long term occupation of network resources caused by long flow, and reduce the flow completion rate while reducing the packet loss rate. But because we use a single queue cache for the switch in Mininet, in order to simulate a shared memory switch. Hedera has packet loss due to switch congestion. So Sonum has the lowest packet loss rate. The traffic distribution of the stag(p, q) model is not concentrated, so under the stage(0.3, 0.3) and stage(0.5, 0.3) models, all three schemes perform well.

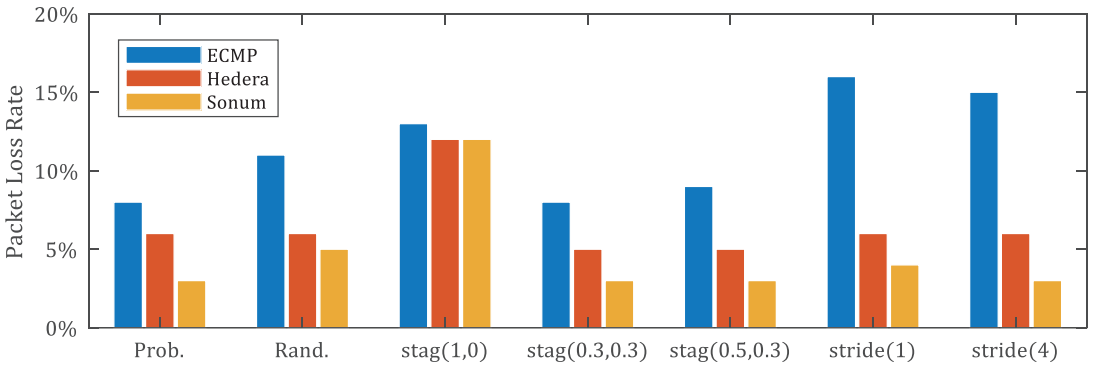

Figure 17. Improvement Performance of long flow FCT of Sonum compared to ECMP and Hedera.

6. Discussion

This paper proposes a synergetic software-defined sampling and optimal network utilization mechanism called Sonum. Sonum divides load balancing in data center networks into two parts: Sampling and scheduling. For sampling, Sonum detects long flows throughout consolidating long flow sampling information from multiple switches, with the result of increasing the exchange of the sampling information and reducing the single switch sampling rate. For scheduling, Sonum minimizes the packet loss rate instead of maximizing the link utilization as the optimization target of load balancing. Sonum translates load balancing into arranging minimum packet loss path for long flows. This paper also introduces a heuristic algorithm to solve scheduling problems. The experimental results show that, compared with ECMP and Hedera, Sonum had a better performance on long flow

detection, network throughput and flow completion time. However, the verification of Sonum is based on the simulation environment, and the deployment effect in the real environment is something to be studied.

Author Contributions: L.T. proposed software-defined synergetic sampling approach and optimal network utilization mechanism, designed the algorithm, and drafted the manuscript. W.S. contributed significantly to manuscript preparation and funding acquisition. L.J. performed the analysis with constructive discussions, and revised this manuscript. P.C. and Z.G. designed and performed the experiment and evalution. In addition, we would like to thank K.Y. and M.L. They participated in the discussion and writing of this manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key R&D Program of China under Grant No. 2018YFB1800305, Fundamental Research Funds for the Central Universities of China under Grant No. 2018YJS002, CERNET Innovation Project under Grant No. NGII20180120 and Jingmen Science and Technology Research and Development Plan Project under Grant No. 2018YFYB055.

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

Abbreviations

The following abbreviations are used in this manuscript:

ICT	Information Communcaiton Technology
DC	Data Center
DCN	Data Center Network
FCT	Flow Completion Time
ECMP	Equal-Cost Multipath
ECN	Explicit Congestion Notification
SJF	Shortest Job First
EWMA	Exponentially Weighted Moving Average
LRU	Link Resource Utilization
SRU	Switch Resource Utilization
NRU	Network Resource Utilization

References

1. Ahmed, H.; Arshad, M.J. Buffer Occupancy-Based Transport to Reduce Flow Completion Time of Short Flows in Data Center Networks. *Symmetry* **2019**, *11*, 646. [CrossRef]
2. Xu, G.; Dai, B.; Huang, B.; Yang, J.; Wen, S. Bandwidth-aware energy efficient flow scheduling with SDN in data center networks. *Future Gener. Comput. Syst.* **2017**, *68*, 163–174. [CrossRef]
3. Chowdhury, M.; Stoica, I. Efficient coflow scheduling without prior knowledge. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 393–406. [CrossRef]
4. Guo, Z.; Hui, S.; Xu, Y.; Chao, H.J. Dynamic flow scheduling for power-efficient data center networks. In Proceedings of the 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), Beijing, China, 20–21 June 2016; pp. 1–10. [CrossRef]
5. Tan, L.Z.; Su, W.; Gao, S.; Cheng, P. L4S: Low-Speed Software Synergetic Sampling and Detecting Long Flow for Data Center Network. In Proceedings of the 2018 International Conference on Networking and Network Applications (NaNA), Xi'an, China, 12–15 October 2018; pp. 169–174. [CrossRef]
6. Sreekumari, P. Multiple Congestion Points and Congestion Reaction Mechanisms for Improving DCTCP Performance in Data Center Networks. *Information* **2018**, *9*, 139. [CrossRef]
7. Benson, T.; Akella, A.; Maltz, D.A. Network Traffic Characteristics of Data Centers in the Wild. In Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, New York, NY, USA, 1–3 November 2010; pp. 267–280. [CrossRef]
8. Zhu, H.; Liao, X.; de Laat, C.; Grosso, P. Joint flow routing-scheduling for energy efficient software defined data center networks: A prototype of energy-aware network management platform. *J. Netw. Comput. Appl.* **2016**, *63*, 110–124. [CrossRef]
9. Luo, S.; Yu, H.; Zhao, Y.; Wang, S.; Yu, S.; Li, L. Towards practical and near-optimal coflow scheduling for data center networks. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 3366–3380. [CrossRef]

10. Cao, Z.; Kodialam, M.; Lakshman, T. Joint static and dynamic traffic scheduling in data center networks. *IEEE/ACM Trans. Netw. (TON)* **2016**, *24*, 1908–1918. [CrossRef]
11. He, K.; Rozner, E.; Agarwal, K.; Felter, W.; Carter, J.; Akella, A. Presto: Edge-based load balancing for fast datacenter networks. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 465–478. [CrossRef]
12. Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. Hedera: Dynamic Flow Scheduling for Data Center Networks. In Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, San Jose, CA, USA, 28–30 April 2010; pp. 19–19.
13. Hu, F.; Hao, Q.; Bao, K. A survey on software-defined network and openflow: From concept to implementation. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 2181–2206. [CrossRef]
14. Zhang, H.; Chen, K.; Bai, W.; Han, D.; Tian, C.; Wang, H.; Guan, H.; Zhang, M. Guaranteeing deadlines for inter-data center transfers. *IEEE/ACM Trans. Netw. (TON)* **2017**, *25*, 579–595. [CrossRef]
15. Zhang, J.; Ren, F.; Tang, L.; Lin, C. Taming TCP incast throughput collapse in data center networks. In Proceedings of the 2013 21st IEEE International Conference on Network Protocols (ICNP), Goettingen, Germany, 7–10 October 2013; pp. 1–10. [CrossRef]
16. Qin, Y.; Yang, W.; Ye, Y.; Shi, Y. Analysis for TCP in data center networks: Outcast and incast. *J. Netw. Comput. Appl.* **2016**, *68*, 140–150. [CrossRef]
17. Anelli, P.; Diana, R.; Lochin, E. FavorQueue: A parameterless active queue management to improve TCP traffic performance. *Comput. Netw.* **2014**, *60*, 171–186. [CrossRef]
18. Benson, T.; Anand, A.; Akella, A.; Zhang, M. Understanding Data Center Traffic Characteristics. *SIGCOMM Comput. Commun. Rev.* **2010**, *40*, 92–99. [CrossRef]
19. Kandula, S.; Sengupta, S.; Greenberg, A.; Patel, P.; Chaiken, R. The Nature of Data Center Traffic: Measurements & Analysis. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, Chicago, IL, USA, 4–6 November 2009; pp. 202–208. [CrossRef]
20. Alizadeh, M.; Greenberg, A.; Maltz, D.A.; Padhye, J.; Patel, P.; Prabhakar, B.; Sengupta, S.; Sridharan, M. Data Center TCP (DCTCP). In Proceedings of the ACM SIGCOMM 2010 Conference, New Delhi, India, 30 August–3 September 2010; pp. 63–74. [CrossRef]
21. Li, Y.; Miao, R.; Kim, C.; Yu, M. Lossradar: Fast detection of lost packets in data center networks. In Proceedings of the 12th International on Conference on Emerging Networking Experiments and Technologies, Irvine, CA, USA, 12–15 December 2016; pp. 481–495. [CrossRef]
22. DeCusatis, C. Optical interconnect networks for data communications. *J. Light. Technol.* **2013**, *32*, 544–552. [CrossRef]
23. Lin, C.Y.; Chen, C.; Chang, J.W.; Chu, Y.H. Elephant flow detection in datacenters using OpenFlow-based Hierarchical Statistics Pulling. In Proceedings of the 2014 IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014; pp. 2264–2269. [CrossRef]
24. Curtis, A.R.; Mogul, J.C.; Tourrilhes, J.; Yalagandula, P.; Sharma, P.; Banerjee, S. DevoFlow: Scaling Flow Management for High-performance Networks. In Proceedings of the ACM SIGCOMM 2011 Conference, Toronto, ON, Canada, 15–19 August 2011; pp. 254–265. [CrossRef]
25. Bi, C.; Luo, X.; Ye, T.; Jin, Y. On precision and scalability of elephant flow detection in data center with SDN. In Proceedings of the 2013 IEEE Globecom Workshops (GC Wkshps), Atlanta, GA, USA, 9–13 December 2013; pp. 1227–1232. [CrossRef]
26. Xing, C.; Ding, K.; Hu, C.; Chen, M. Sample and fetch-based large flow detection mechanism in software defined networks. *IEEE Commun. Lett.* **2016**, *20*, 1764–1767. [CrossRef]
27. Xu, H.; Li, B. TinyFlow: Breaking elephants down into mice in data center networks. In Proceedings of the 2014 IEEE 20th International Workshop on Local & Metropolitan Area Networks (LANMAN), Reno, NV, USA, 21–23 May 2014; pp. 1–6. [CrossRef]
28. Kabbani, A.; Vamanan, B.; Hasan, J.; Duchene, F. Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, Sydney, Australia, 2–5 December 2014; pp. 149–160. [CrossRef]
29. Chakraborty, S.; Chen, C. A low-latency multipath routing without elephant flow detection for data centers. In Proceedings of the 2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR), Yokohama, Japan, 14–17 June 2016; pp. 49–54. [CrossRef]

30. Carpio, F.; Engelmann, A.; Jukan, A. DiffFlow: Differentiating short and long flows for load balancing in data center networks. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6. [CrossRef]
31. Benson, T.; Anand, A.; Akella, A.; Zhang, M. MicroTE: Fine grained traffic engineering for data centers. In Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies, Tokyo, Japan, 6–9 December 2011; p. 8. [CrossRef]
32. Suh, J.; Kwon, T.T.; Dixon, C.; Felter, W.; Carter, J. OpenSample: A low-latency, sampling-based measurement platform for commodity SDN. In Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems (ICDCS), Madrid, Spain, 30 June–3 July 2014; pp. 228–237. [CrossRef]
33. Afek, Y.; Bremler-Barr, A.; Feibish, S.L.; Schiff, L. Detecting heavy flows in the SDN match and action model. *Comput. Netw.* **2018**, *136*, 1–12. [CrossRef]
34. Wang, W.; Sun, Y.; Salamatian, K.; Li, Z. Adaptive path isolation for elephant and mice flows by exploiting path diversity in datacenters. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 5–18. [CrossRef]
35. Bai, W.; Chen, L.; Chen, K.; Han, D.; Tian, C.; Sun, W. PIAS: Practical information-agnostic flow scheduling for data center networks. In Proceedings of the 13th ACM Workshop on Hot Topics in Networks, Los Angeles, CA, USA, 27–28 October 2014; pp. 25:1–25:7. [CrossRef]
36. Tan, L.; Su, W.; Gao, S.; Miao, J.; Cheng, Y.; Cheng, P. Path-flow matching: Two-sided matching and multiobjective evolutionary algorithm for traffic scheduling in cloud date center network. *Trans. Emerg. Telecommun. Technol.* **2019**, e3809. [CrossRef]
37. Zhang, J.; Ren, F.; Yue, X.; Shu, R.; Lin, C. Sharing bandwidth by allocating switch buffer in data center networks. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 39–51. [CrossRef]
38. Zhang, Y.; Cui, L.; Zhang, Y. A stable matching based elephant flow scheduling algorithm in data center networks. *Comput. Netw.* **2017**, *120*, 186–197. [CrossRef]
39. Curtis, A.R.; Kim, W.; Yalagandula, P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In Proceedings of the International Conference on Computer Communications, Shanghai, China, 10–15 April 2011; Volume 11, pp. 1629–1637. [CrossRef]
40. Chen, L.; Lingys, J.; Chen, K.; Liu, F. Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, Budapest, Hungary, 20–25 August 2018; pp. 191–205. [CrossRef]
41. Lucas, J.M.; Saccucci, M.S. Exponentially weighted moving average control schemes: Properties and enhancements. *Technometrics* **1990**, *32*, 1–12. [CrossRef]
42. Liu, Z.; Gao, D.; Liu, Y.; Zhang, H.; Chao, H.C. Optimizing Elephant Flow Sch3yuefen eduling for SDN-Based Data Center Network. *J. Internet Technol.* **2017**, *18*, 1–9. [CrossRef]
43. Tan, L.Z.; Tan, Y.Y.; Yun, G.X.; Zhang, C. An improved genetic algorithm based on k-means clustering for solving traveling salesman problem. In Proceedings of the 2016 International Conference on Computer Science, Technology and Application (CSTA2016), Changsha, China, 18–20 March 2016; pp. 334–343. [CrossRef]