symmetry

MDPI

Article

Novel Multi-Level Dynamic Traffic Load-Balancing Protocol for Data Center

Sheeba Memon [1], **Jiawei Huang** [1,*], **Hussain Saajid** [2], **Naadiya Khuda Bux** [1], **Arshad Saleem** [3] and **Yazan Aljeroudi** [4]

[1] School of Information Science and Engineering, Central South University Changsha, Changsha 410083, China; sheeba.memon@csu.edu.cn (S.M.); naadiya.khudabux@csu.edu.cn (N.K.B.)
[2] School of Computer Science and Engineering, Dalian University of Technology Dalian, Dalian 116024, China; sunny_sau@hotmail.com
[3] Shaheed Benazir Bhutto University, Shaheed Benazirabad, Landhi-Nawabshah 67450, Pakistan; arshadsaleem1968@gmail.com
[4] Mechanical Engineering Department, International Islamic University Malaysia, Kuala Lumpur 53100, Malaysia; yazan.aljeroudi@gmail.com
* Correspondence: jiaweihuang@csu.edu.cn; Tel.: +86-0731-8883-0212

Received: 17 December 2018; Accepted: 21 January 2019; Published: 28 January 2019

Abstract: Typically, the production data centers function with various risk factors, such as for instance the network dynamicity, topological asymmetry, and switch failures. Hence, the load-balancing schemes should consider the sensing accurate path circumstances as well as the reduction of failures. However, under dynamic traffic, current load-balancing schemes use the fixed parameter setting, resulting in suboptimal performances. Therefore, we propose a multi-level dynamic traffic load-balancing (MDTLB) protocol, which uses an adaptive approach of parameter setting. The simulation results show that the MDTLB outperforms the state-of-the-art schemes in terms of both the flow completion time and throughput in typical data center applications.

Keywords: data centers; load balance; parameter setting

1. Introduction

The exponential growth of the information technology (IT) industry and its storage demand forces to move toward cloud computing, in which a huge amount of computing and storage resources are provided in large data centers [1,2]. These data centers are designed with high-speed, highly configured servers. The cloud servers are used to offer web searches, Amazon Elastic Compute Cloud (EC2), Microsoft azure storage, and personal and business storage [3–5]. As a consequence, the cloud servers handle a huge amount of data, and the interconnected networks and server resources eventually become unstable, which hits the performance of the network bandwidth [6]. Due to this, the network bandwidth, the latency-sensitive web services, and applications are suffering from a long network latency tail [7]. An illustrating image of the data center traffic load is shown in Figure 1.

The multi-rooted topologies have been adopted for data center networks. This is to provide the high bisection bandwidth [8]. The existing multipath topologies are used to deliver the alternative routing paths between any two end-hosts under different switches.

The load balancing for multiple paths with the utilization of whole network resources can improve throughput as well as reduce latency for data center applications [9]. The production data centers are used to operate under several uncertainty factors. The uncertainty factors are mainly the dynamicity of the traffics, unstable links, device heterogeneity, and switch failures [10]. To overcome these situations due to uncertain circumstances, the load-balancing technique is seriously effective.

Using the load-balancing technique, the accuracy of the path-sensing conditions and path rerouting according to the path conditions can be improved. However, many of these protocols are unaffected by path conditions; the approach is to split blindly at a fixed granularity. The proposal was made by ignoring traffic conditions; however, it can be seen that it affects the performance, especially when asymmetry arises [7,11]. The flowlet-based proposals have been proposed to sense the path conditions. However, it facilitates rerouting only when the flowlet emerges. The flowlet formation depends on various factors that are related to the applications and the transport layer. As a result, using a flowlet-based reaction to the path conditions does not provide a timely reaction to congestion [10]. In addition, switch modification-based approaches (CONGA and HULA) are based on switching or hardware modification that imposes impracticalities [12,13]. Among other solutions, Hermes has been impressing through its better performance [14]. Hermes accomplishes per-flow load balancing to enact faster rerouting following the network status. It applies packets to react timely to overcome uncertainty challenges. However, it also imposes the congestion mismatch and packet reordering as well; in such conditions, Hermes reduces the frequency of rerouting to avoid congestion. Moreover, Hermes uses the unsystematic hashing, round robin, and seeks the best path to route and reroute based on the path conditions as well as flow status. However, stability is a key concern for the condition when flows interact in the network [10]. The bursty network congestion has been well addressed in [15–17] at the transport and link layers. Their performances can be further improved with the support from the load balancing schemes at network layer.

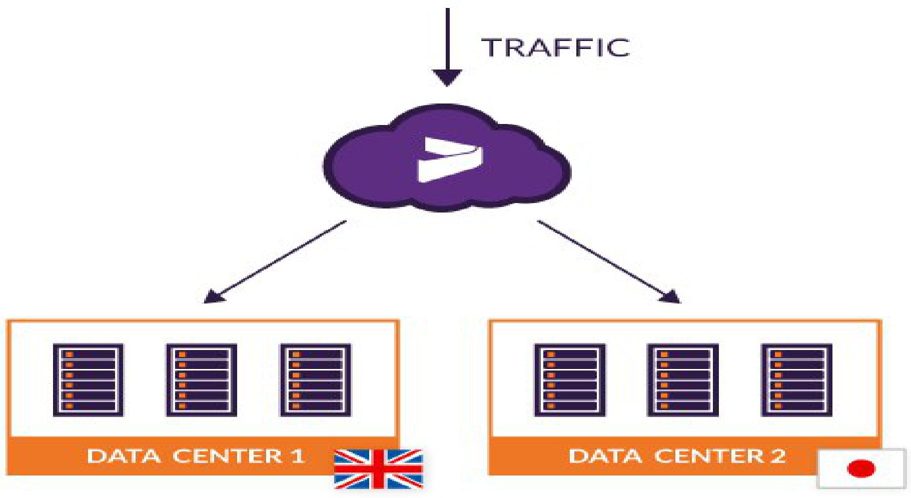


Figure 1. Data center image of application.

The main contribution of this article is a multi-level dynamic traffic load-balancing (MDTLB) protocol. The key idea is to design a protocol using the functionalities of Hermes, multiple levels of path characterization, and an adaptive approach to setting parameters. Two algorithms are proposed, namely, the judging algorithm and the rerouting algorithm.

The rest of the article is organized as follows. Section 2 discusses the background study, and Section 3 provides the motivation of the study. Section 4 discusses the methodology of the proposed approach. Section 5 presents the result and discussion of the proposed approach, while Section 6 demonstrates the related works. Finally, Section 7 contains the findings and the conclusion.

2. Background

The advancement of data center and virtualization technologies has led cloud storage. The concept of the cloud computing paradigm delivers through online network accesses shared resources that are configurable as well as flexible resources. Thereby, load-balancing protocol and its approaches are significant in cloud servers. The load-balancing protocols are mainly designed with the consideration of three types of flows (static, dynamic proactive, and dynamic reactive) and the granularity [7]. The static flow is then allocated to the available paths through applying the fixed conditions by the use of the hash for each packet header. Conversely, this technique is inflexible to the scenarios of throughput-oriented flows, while it is consigned to the same path. In such a situation, the flows cannot move to other less utilized paths. Meanwhile, the dynamic reactive flow is able to move across any of the available paths subject to available bandwidth. The dynamic reactive flow provides a significant result. However, it increases higher complexity in measuring link utilization, flow counting, and estimating the best flow. On the other hand, the dynamic proactive allows the flows to become assigned to the available paths in a fixed assignment, where the initial assignment is mainly performed according to the available bandwidth. This approach is better compared to static flow and dynamic reactive flow in terms of implementation overhead, flexibility, and performance. Four kinds of granularity are observed such as per packet, per flow, flowlets, and per flow cell.

- Per packet: Can be used for the best load balance; however, due to the higher reordering, there are chances to attain insignificant reordering.
- Flowlets: Due to the variation of the candidate path latencies, the flowlet's size changes enthusiastically. If the latency rates are higher, the flowlets become large. It is obvious that it applies per packet and per flow technique. Therefore, the load balancing is then fine as well as coarse-grained. As a result, the flowlets become promising in load balancing for asymmetric paths. Nevertheless, the flowlets causes a small flow reordering, which can interrupt the execution times.
- Per flow cell: It uses the fixed size with the consideration of tens of packets. The most positive effect of per flow cell is that it uses simplified load balancing to reduce the possible reordering of small flows. However, it rises the reordering for larger flows, which can be fragmented into several flow cells.

3. Motivation of the Study

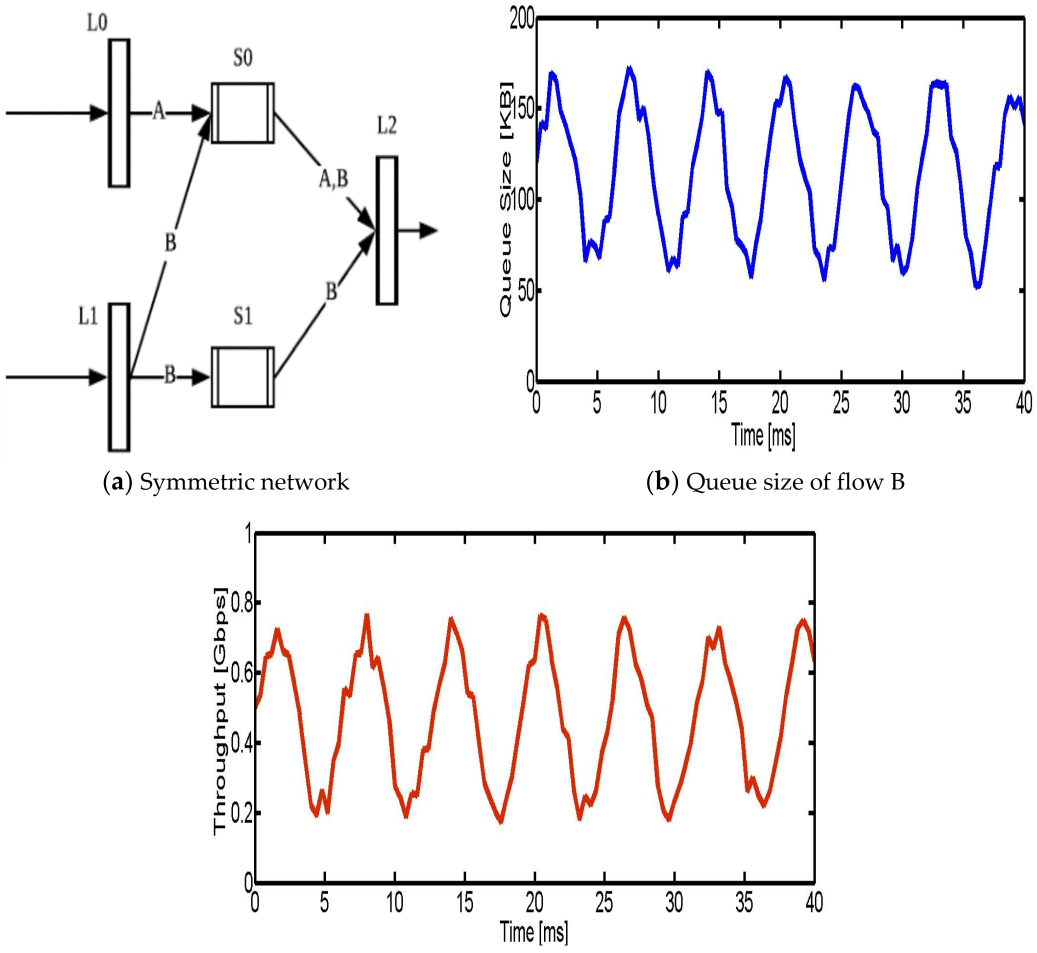
This study is focused on improving the performance of the load balancer to avoid the problem of congestion mismatch. This section discusses the concept of the congestion mismatch.

3.1. Congestion Mismatch Problem with an Example

To avoid the congestion mismatch issues, the algorithm may consider the flow-based rate of the congestion state using the condition of the paths and the routing. Regarding the event of rerouting, there are chances of mismatches. This is happening because of the sending rate and the rerouting state of the new path. The dynamic rerouting within a flow allows the various congestion states of the paths to adjust the rate of newer paths. This occurrence is then stated as congestion mismatch [14]. To illustrate the consequences of the congestion mismatch problem, two scenarios are considered. The first scenario is considered under asymmetric topologies for the same flow sizes, A and B, and the second scenario is for a heterogeneous network. The second scenario presents the load distributions according to link capacities. Figure 2 depicts the dynamic routing protocols of Presto and Digit-Reversal Bouncing (DRB) [7].

In Figure 2a we have a simple three to two leaf-spine topology with a broken link from L0 to S1. Also, we have two types of flows: flow A, which is a data center transmission control protocol (DCTCP) from L0 to L2, and flow B, which is a user datagram protocol (UDP) flow from L1 to L2. The result of congestion mismatch is low throughput and queue oscillations when equal weights are adopted for different paths. The throughput is oscillating around 1 Gbps, as shown in Figure 2c, while the queue size is around 100 KB, as shown in Figure 2b. This is interpreted by the vigorous rerouting; the congestion feedback (i.e., explicit congestion notification (ECN)) of the upper path constrains the congestion window, resulting in throughput loss in the bottom path. Furthermore, when flow B with a larger window shifts from the bottom path to the upper path, the upper path cannot immediately absorb such a burst, causing queue length oscillations.

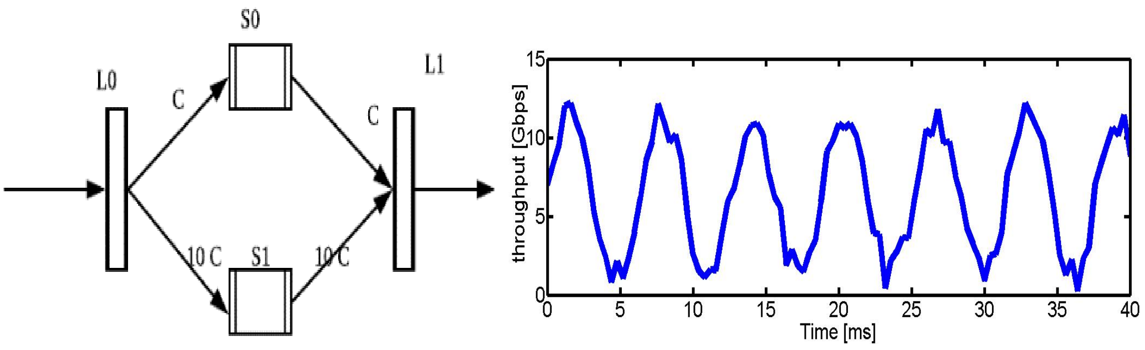
A variation in the size of the queues appears in the S0–L2 link, whereas the low throughput can be measured in flow B. Furthermore, observing the distribution of loads according to capacities is shown in Figure 3. In the S0–L1 link, L1 suffers from low throughput, and the S0–L1 link has a limitation regarding queue size variations.



(**a**) Symmetric network (**b**) Queue size of flow B

(**c**) Throughput of flow B

Figure 2. Asymmetry because of broken link between L0 and S1, when the link capacity is 10 G [7].



(**a**) Hybrid network (**b**) Throughput of flow A

Figure 3. Different capacities of paths between the L0 S0 and L0 S1 links [7].

In addition, we show that congestion mismatch remains harmful even if we distribute the flows proportionally to path capacity. To illustrate this, we consider a heterogeneous network with 1 C and 10 C paths, as shown in Figure 3a. We spread the flows using a 1:10 ratio to match the path capacities, and expect both paths to be fully utilized.

However, as shown in Figure 3b, flow A can only achieve an overall throughput of around five Gbps. To understand the reason, assume that the first 10 flow cells go through the 10 C path. Since the path is not congested, the protocol will increase the congestion window without realizing that the subsequent flow cell will go through the 1 C path. With a large congestion window, the 11th flow cell is sent at a higher rate on the 1 C path, causing a rapid queue buildup on the output port of spine S0 to leaf L2. As the queue length exceeds the ECN marking threshold, the DCTCP will reduce the window upon receiving ECN-marked ACKs without realizing that such a reduction will affect the following flow cells on the 10 C path. As a result, such a congestion mismatch still causes throughput loss and queue oscillations.

3.2. Summary

Our analysis leads us to conclude that the congestion mismatch problem occurs due to the continuous path rerouting, which is happening because of the sending rate and the rerouting state of the new path and the dynamic rerouting within a flow that allows the various congestion states of the paths to adjust the rate of newer paths. These outcomes inspire us to handle the congestion mismatch problem by reducing rerouting to the minimum. We overcome the mismatching problem that is proved by improving the flow completion time (FCT) and the throughput, and we implemented our scheme to not reroute until it is a critical situation such as failure or very bad paths, so that the rerouting is minimized as much as possible.

4. Proposed Multi-Level Traffic Load Balancing MDTLB

4.1. Basic Concept

The existing Hermes applies the thresholds and parameters to depict the path, identify the best possible path through the confirmation of the lower round trip time (RTT) measurement, and lower the rate of congestion mismatch. Hence, the congested path can be identified while there are higher RTT and ECN. It is obvious that the higher RTT can be lowered due to the stack latency of the network, and the higher ECN fraction comes from some of the inaccurate samples. Else, the path is stated to be gray in all of the other cases. This characterization approach is sensitive to the parameters setting. It causes a non-steady performance that may increase the congestion mismatch effect on the performance instead of decreasing it due to the wrong threshold values. However, a multi-level path judging with dynamic thresholds-based heuristic characterization of the path is more realistic regarding reducing a congestion mismatch issue than a simple threshold-based approach.

4.2. Design Considerations

The design of the protocol is composed of two parts: an adaptive parameter-setting algorithm, and a rerouting algorithm. The detailed discussion is as below:

4.2.1. Parameters

The parameters setting for flow-related variables parameters is shown in Table 1, the path level variables are presented in Table 2, and the parameters for the developed protocol are provided in Table 3.

Table 1. Flow Level Variables.

Flow Level Variable	Meaning
$if_{timeout}$	Set if the flow experiences a timeout
S_{sent}	Size sent from the flow
$n_{timeout}$	Number of times out that the flow experiences
r_{sent}	Sending rate of the flow

Table 2. Path Level Variables.

Path level variable	Meaning
$n_{timeout}$	Number of timeout events of path
$f_{retransmission}$	Fraction of retransmission events of path
$Type$	Path condition

Table 3. Path Level Variables.

$\boldsymbol{T_{ECN}}$	**Threshold for Fraction of ECN**
T_{RTT_LowS}	Static Threshold for low RTT,
T_{RTT_highS}	Static Threshold for high RTT
T_{RTT_LowD}	Dynamic Threshold for low RTT
T_{RTT_highD}	Dynamic Threshold for high RTT
Δ_{RTT}	Threshold for notably better RTT
Δ_{ECN}	Threshold for notably better ECN fraction

4.2.2. Adaptive Parameters Settings

To overcome the parameters setting sensitivity of Hermes, this article proposed two steps of improvement:

1. The first is to increase the number of path levels to give the scheme more awareness of the path's status. Hence, the path is divided into five types {very good, good, gray, bad, and very bad}, as shown in Figure 4.
2. The second step is to make the judging dynamic by introducing static and dynamic T_{RTT} thresholds, as shown in Figure 5 and pseudocode one. The user sets the static thresholds, while the dynamic thresholds are set by the system using an adaptive way to determine the best threshold value based on the network load. It is important to note that T_{RTT} can be used as the main judging thresholds. This is because the measurements of T_{RTT} are more accurate and can give a wider range than T_{ECN} to distinguish between path levels.

The judging algorithm is illustrated in pseudocode one and Figure 6.

```
1. for each path p do:
2. if $f_{ECN} < T_{ECN}$ and $T_{RTT} < T_{RTT\_LowS}$ then type = very good
3. else if $f_{ECN} < T_{ECN}$ and $T_{RTT} < \boldsymbol{T_{RTT\_LowD}}$ then type = good
4. else if $f_{ECN} > T_{ECN}$ and $T_{RTT} > T_{RTT\_highS}$ then type = bad
5. else if $f_{ECN} > T_{ECN}$ and $T_{RTT} > \boldsymbol{T_{RTT\_highD}}$ then type = very bad
6. else type = gray
7. if ($n_{timeout} > 3$ and no packet is ACKed) or ($f_{retransmission} > 1\%$ and type $\neq$ bad or very bad), then type =
failed
end for
```

```
If (good portion is small) or (good portion is medium and bad portion is big) or (good portion is big and bad portion is big) then changeRatio = ChangeRatio + 10
Else if ( good portion is big and bad portion is not big ) then changeRatio = ChangeRatio − 10
End
```

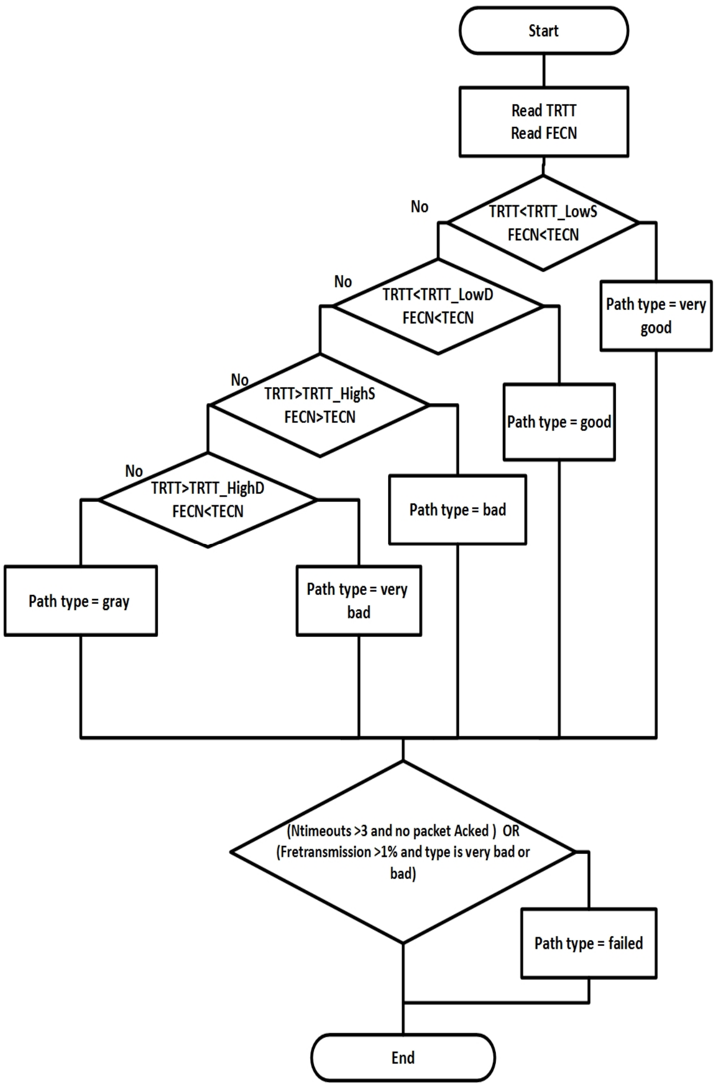


Figure 4. Block diagram of path judging logic.

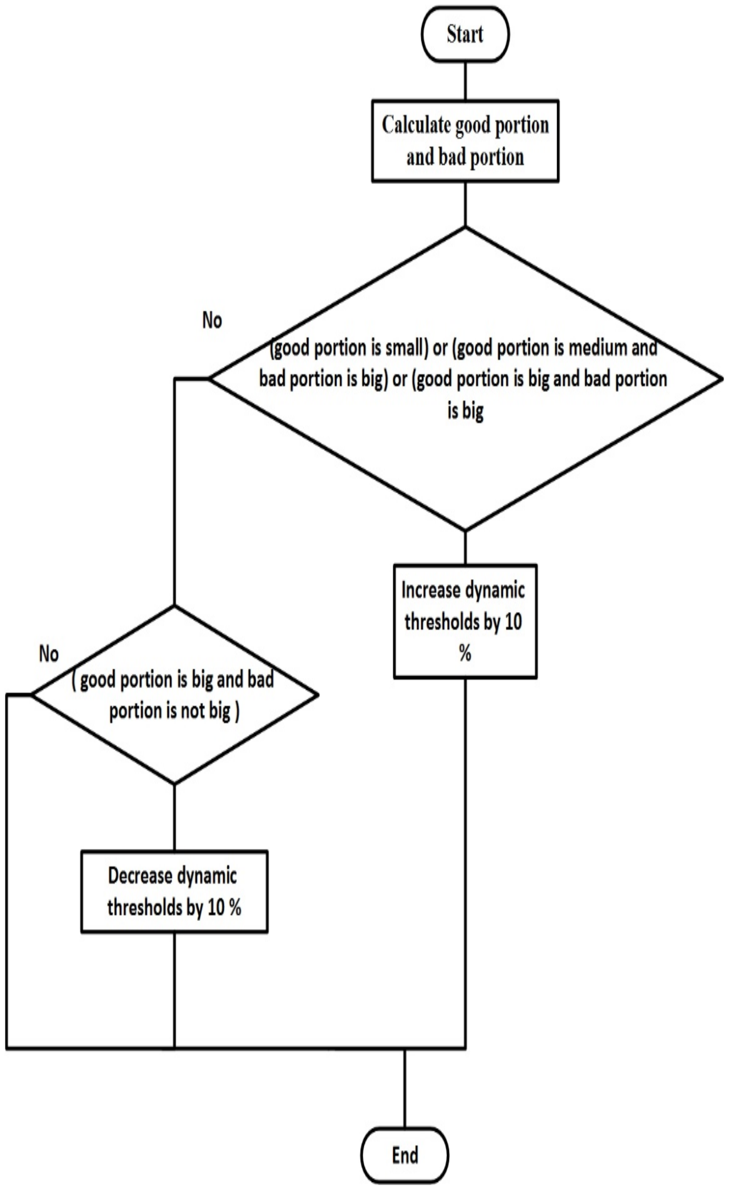

Start
Calculate good portion and bad portion
(good portion is small) or (good portion is medium and bad portion is big) or (good portion is big and bad portion is big
No
Increase dynamic thresholds by 10 %
(good portion is big and bad portion is not big)
No
Decrease dynamic thresholds by 10 %
End

Figure 5. Block diagram of dynamic threshold logic.

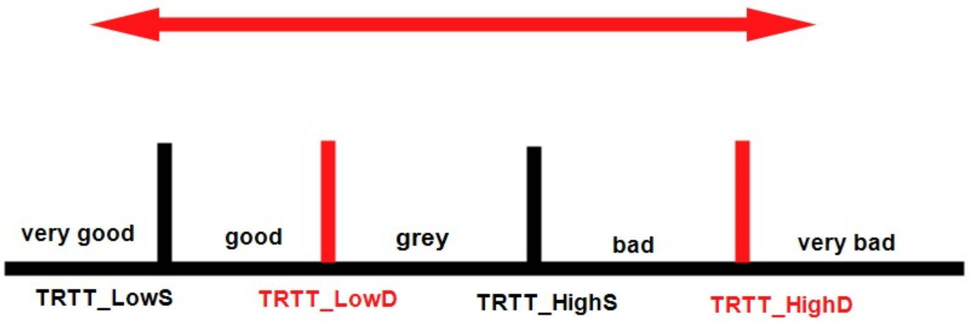


Figure 6. Multi-level path judging.

4.2.3. Rerouting Logic

The proposed algorithm for rerouting logic is designed that will find the best available path for every flow. The idea of the proposed logic can be derived as: if there is new flow, then, the process is to begin to search for very good paths; if the search result does not find any (none), then the process searches for gray if none chose the path randomly. In such cases, the proposed approach chose a path with the smallest local sending rate from the available paths. If there is old flow with a very bad path with a small rate and the sent size is less than size threshold S, then the approach is applied to find a better path from the very good, good, or gray. If there is no better path than the current one, it leaves and cancels the rerouting process. An illustration can be seen in Figure 7.

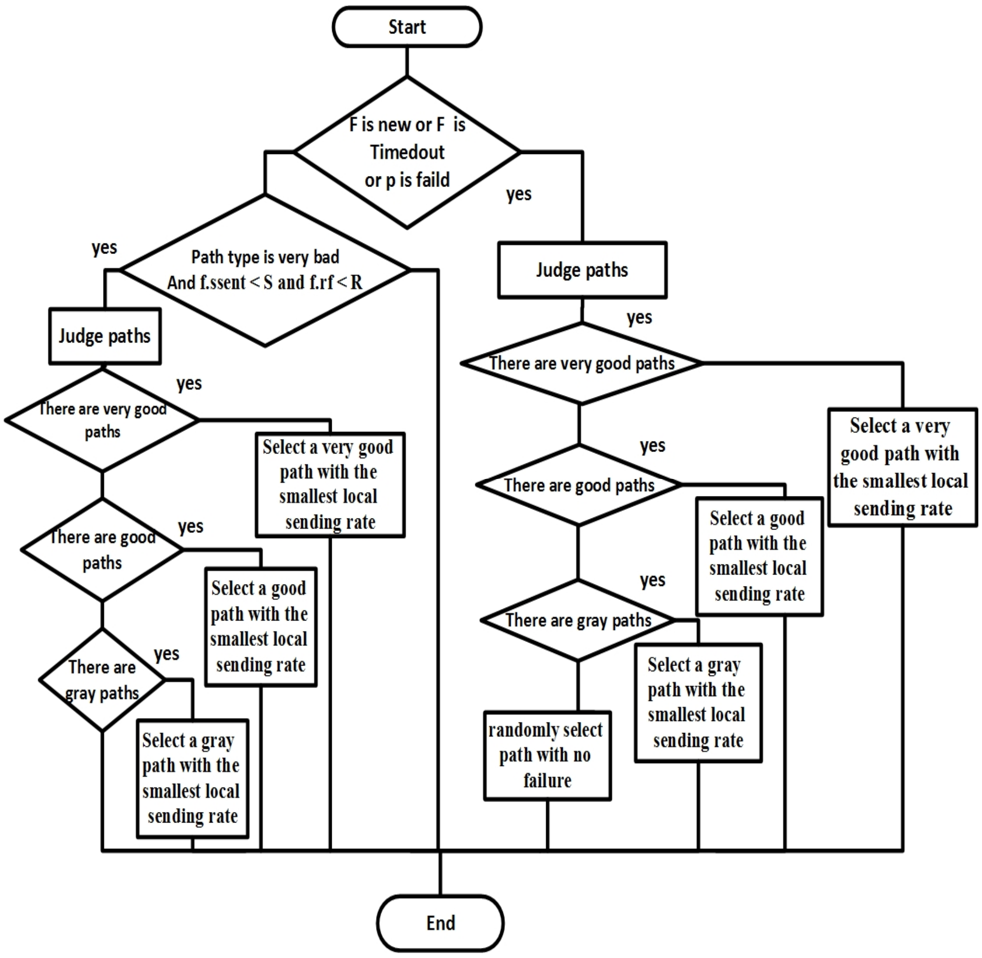

Start
F is new or F is Timedout or p is faild
yes
Path type is very bad And f.ssent < S and f.rf < R
yes
Judge paths
There are very good paths
yes
Select a very good path with the smallest local sending rate
There are good paths
yes
Select a good path with the smallest local sending rate
There are gray paths
yes
Select a gray path with the smallest local sending rate
Judge paths
yes
There are very good paths
Select a very good path with the smallest local sending rate
yes
There are good paths
Select a good path with the smallest local sending rate
yes
There are gray paths
Select a gray path with the smallest local sending rate
randomly select path with no failure
End

Figure 7. Rerouting logic.

The logic of rerouting is illustrated in pseudocode two.

```
1.  for every packet do
2.  Assume its corresponding flow is f and path is p
3.  if f is a new flow or $f.if_{timeout}$ ==true or $p_{type}$ == failed then
4.  {p′} = all very good paths
5.    if {p′} $\neq \Phi$ then
6.        P* = $Argmin_{p \in \{p'\}}(p.r_p)$
      /* Select a very good path with the smallest local sending rate/*
7.    else
8.      {p"} = all good paths
9.      if {p"} $\neq \Phi$ then
10.         P* = $Argmin_{p \in \{p''\}}(p.r_p)$
      /* Select a good path with the smallest local sending rate */
          else
11.                 {p"′} = all gray paths
12.         if {p"′} $\neq \Phi$ then
13.                 p* = $Argmin_{p \in \{p'''\}}(p.rp)$
14.         else
15.         p* = a randomly selected path with no failure
16.  else if p:type == very bad then
17.      if f.ssent < S and f.rf < R then
18.      {P′} = all very good paths notably better than p
19.      / $\forall p' \in \{P'\}$, we have p.tRTT – p′.tRTT > $\Delta_{RTT}$ and p.fECN – p′.fECN > $\Delta_{ECN}$ */
20.           if {p′} $\neq \Phi$ then
21.            P* = $Argmin_{p \in \{p'\}}(p.rp)$
22. else
23.  {p"} = all good paths notably better than p
24.  if {p"} $\neq \Phi$ then
25.      P* = $Argmin_{p \in \{p''\}}(p.rp)$
26.        /* Select a good path with the smallest local sending rate */
27.          else
28.               {p"′} = all gray paths notably better than p
29.      if {p"′} $\neq \Phi$ then
30.               p* = Argmin $_{p \in \{p'''\}}(p.rp)$
31.         else
32.           p = p /* Do not reroute */
33.    return p* /* The new routing path */
```

4.2.4. Example of Proposed Scheme

Suppose we have two paths, P1 and P2, which both have the capacity of 10 G, and were received for flows A, B, C, D, E, F, G, H, and I. As illustrated in Figure 8 and Table 4, the following events describe the system response for this scenario:

1. Flow A started; the status of paths is unknown, so they are grey; the portion of good paths equals the portion of bad paths (0), so no need to change the dynamic thresholds, and our routing logic will choose any of the paths, let us say P1.
2. Flow B started; the judging of paths will be bad for P1 due to traffic and grey for P2; the routing logic will choose P2; the portion of bad paths is greater than the portion of good paths, so the dynamic threshold should be increased by 10%.
3. Flow C started and due to the change of the dynamic threshold from the previous step, the judging logic will be good for P1 and bad for P2; as a result, the routing logic will choose P1, and the portion of good paths equals the portion of bad paths, so no need to change the dynamic thresholds.

4. Flow D started, the judging logic will be bad for P1 and P2; the routing logic will choose either P1 or P2, let us say P1; the portion of bad paths is greater than the portion of good paths, so the dynamic thresholds should be increased by 10%.
5. Flow E started, and due to the change of dynamic thresholds from the previous step, the judging logic will be very bad for P1 and good for P2; the routing logic will choose P2, and the portion of good paths equals the portion of bad paths, so no need to change the dynamic thresholds.
6. Flow F started, the judging logic will be very bad for P1 and bad for P2; the routing logic will choose P2, the portion of bad paths is greater than the portion of good paths, so the dynamic thresholds should be increased by 10%.
7. Flow G started, and due to the change of dynamic threshold from the previous step, the judging logic will be very good for P1 and good for P2; the routing logic will choose P1, and the portion of bad paths is smaller than the portion of good paths, so the dynamic thresholds should decrease by 10%.
8. Flow H started, and due to the change of dynamic threshold from the previous step, the judging logic will be good for P1 and good for P2; the routing logic will chose either P1 or P2, let us say P2; the portion of bad paths is smaller than the portion of good paths, so the dynamic thresholds should decrease by 10%.
9. Flow I started, and the judging logic will be good for P1 and bad for P2; thus, the routing logic will choose P1, and the portion of good paths equals to the portion of bad paths, so no need to change the dynamic thresholds.

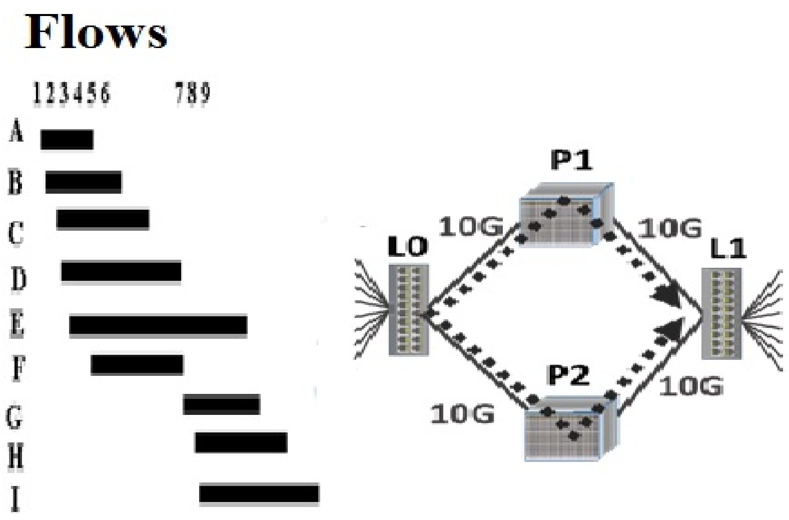


Figure 8. The example of the multi-level dynamic traffic load-balancing (MDTLB) method.

Table 4. Example summary.

Events	1	2	3	4	5	6	7	8	9
Flow	A	B	C	D	E	F	G	H	I
P1 type	Grey	Bad	Good	Bad	Very Bad	Very Bad	Very Good	Good	Good
P2 type	Grey	Gray	Bad	Bad	Good	Bad	Good	Good	Bad
chosen path	P1	P2	P1	P1	P2	P2	P1	P2	P1
Dynamic thresholds	No change	+10%	No change	+10%	No change	+10%	−10%	−10%	No change

5. Results and Discussion

The proposed algorithm was implemented in a Network Simulator 3 (NS3) environment. The simulation environment was configured using the selected parameters. The simulation parameters are listed in Table 5. We tested MDTLB under a symmetric 4 × 4 leaf-spine topology, as shown in

Figure 9, with 64 hosts connected by 10 Gbps links. We simulate a 2:1 oversubscription at the leaf level similar to the typical data center deployments.

Table 5. Simulation Parameters.

Values	Parameters
runMode	MDTLB, TLB (Hermes), Conga, Presto, DRB, ECMP, Clove, DRILL, LetFlow
transportProt	Tcp
enableLargeDupAck	False
enableLargeSynRetries	False
enableFastReConnection	False
enableLargeDataRetries	False
For each leaf serverCount	8
spineCount	4
leafCount	4
linkCount	1
spineLeafCapacity	10
leafServerCapacity	10 Gbps
linkLatency	10 us
cdfFileName	data mining: VL2_CDF, web search: DCTCP_CDF, general flows: (VL2_DF + DCTCP_CDF)/2
load	[0.3, 0.5, 0.7, 0.9]
MDTLBMinRTTS	40 us
MDTLBHighRTTS	180 us
Initial MDTLBMinRTTD	40 us
Initial MDTLBHighRTTD	180 us
MDTLBBetterPathRTT	1 us
MDTLBT1	100 us
MDTLBECNPortionLow	0.1
MDTLBProbingEnable	True
MDTLBProbingInterval	50 us
MDTLBSmooth	True
MDTLBRerouting	True
MDTLBS	64,000 byte
MDTLBReverseACK	True
quantifyRTTBase	10
MDTLBFlowletTimeout	5 ms

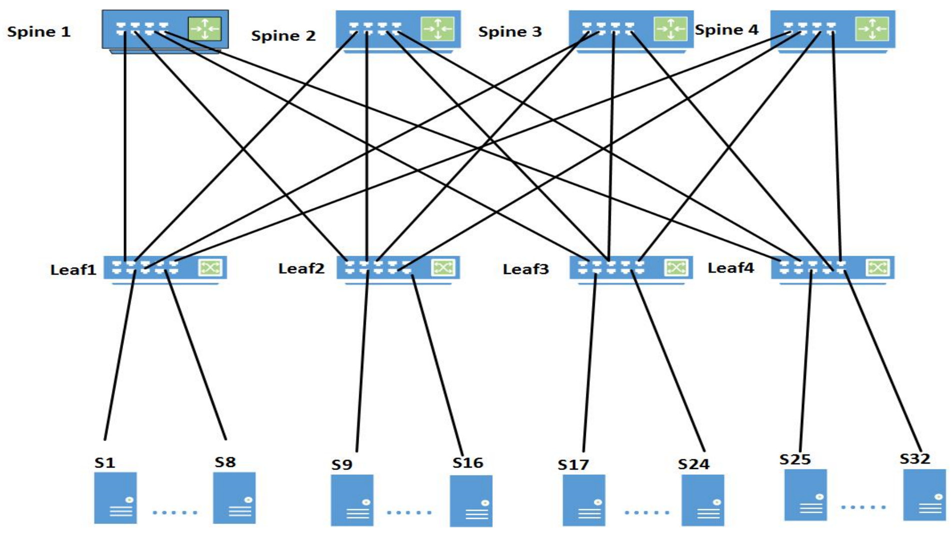
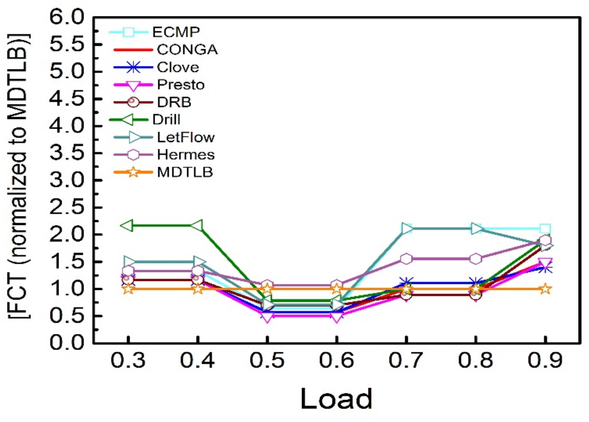
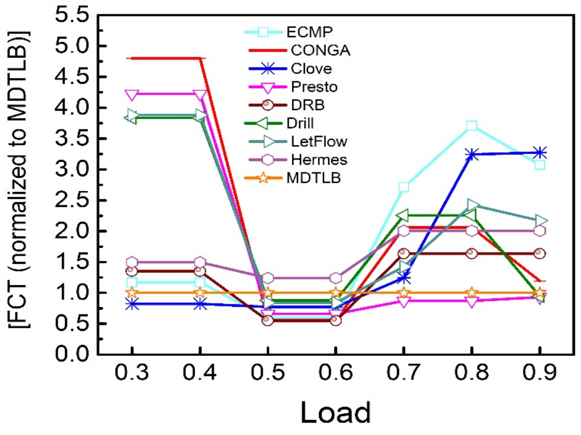


Figure 9. Leaf-Spine topology.

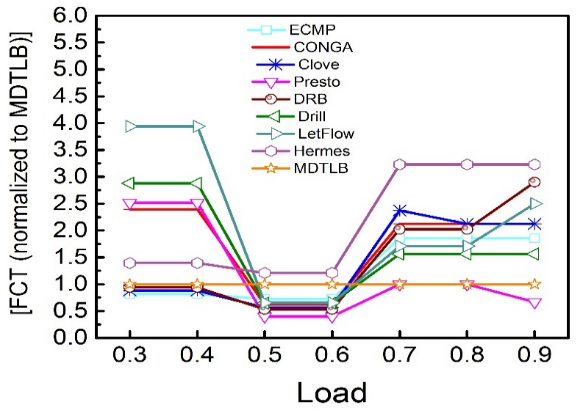
We evaluate and measure the performance of the proposed MDTLB by two main metrics: the (normalized) flow completion time (FCT) and the throughput. To do so, general flow, data mining, and the web search workload were used. The observation of the FCT of general flows, with three types of flows, are also shown separately for short flows, long flows, and both short and long flows in Figure 10. Similarly, in Figures 11 and 12, the observation was made for the flow completion time (FCT) of web search and data-mining, respectively. In all of the figures of FCT, the results were normalized to MDTLB protocol in order to have better visualization for comparing results.



(**a**) Short Flows FCT

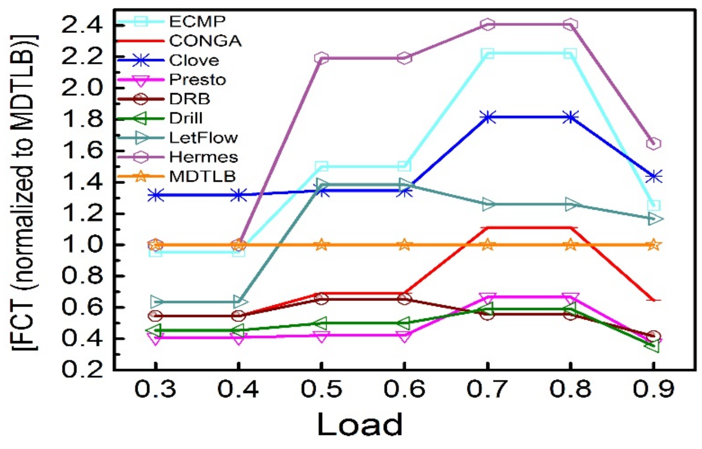


(**b**) Long flows FCT

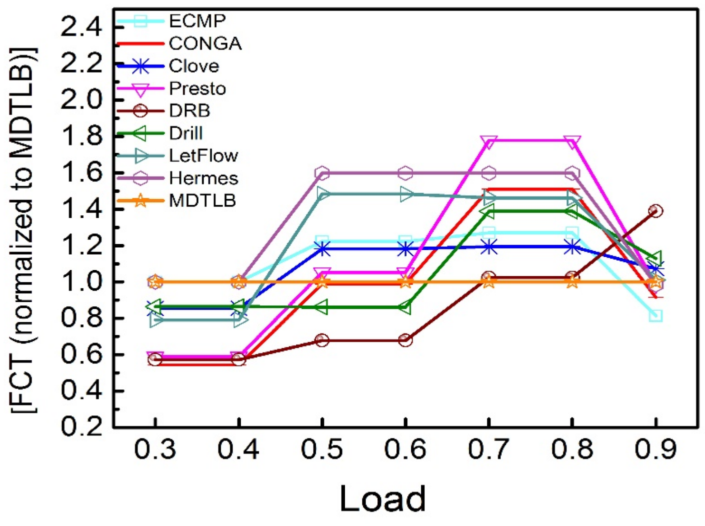


(**c**) Overall FCT of both short and long flows

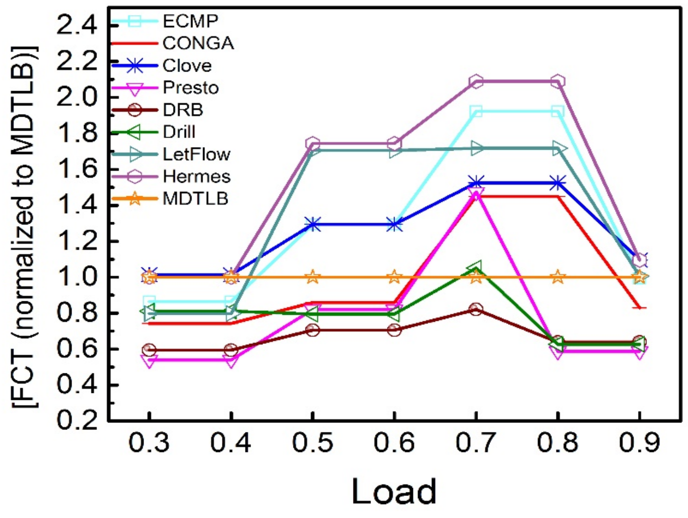
Figure 10. Flow completion time of various loads for the general flows in the symmetric topology with three types of flows: (**a**) short flows FCT; (**b**) long flows FCT; (**c**) and overall FCT of both short and long flows.



(**a**) Short flows FCT

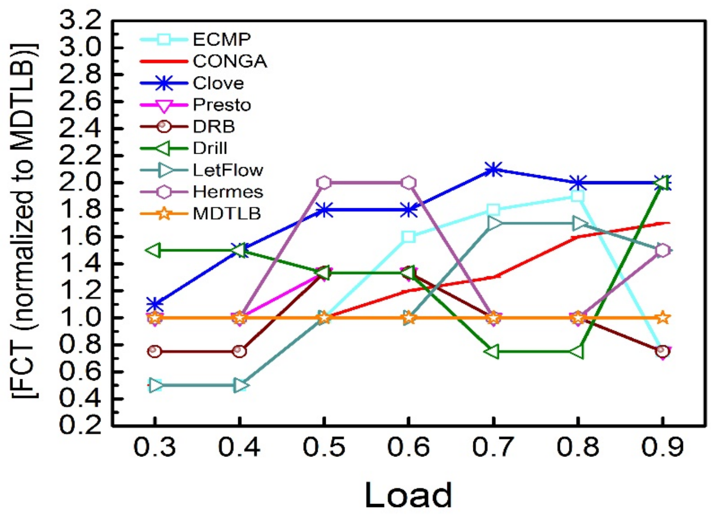


(**b**) Long flows FCT

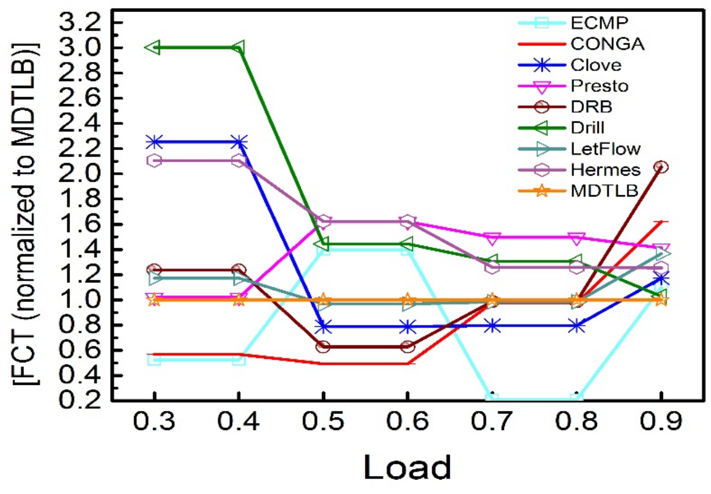


(**c**) Overall FCT of both short and long flows

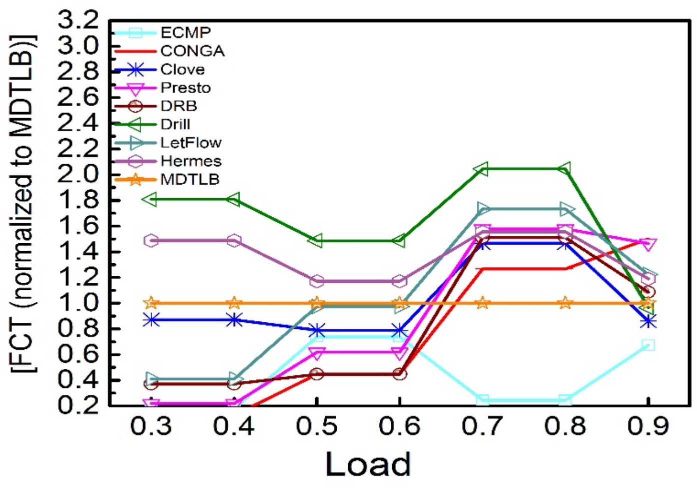
Figure 11. Flow completion time of various loads for the web search application in the symmetric topology with three types of flows: (**a**) short flows FCT; (**b**) long flows FCT; (**c**) and overall FCT of both short and long flows.



(**a**) Short flows FCT



(**b**) Long flows FCT



(**c**) Overall FCT of both short and long flows.

Figure 12. Flow completion time of various loads for the data-mining application in the symmetric topology with three types of flows: (**a**) short flows FCT; (**b**) long flows FCT; (**c**) and overall FCT of both short and long Flows.

From the above results, the following observation can be made:

Under general workload

As we observe in Figure 10, we use the FCT of short flows, long flows, and a combination of both short and long flows with general flows to compare the MDTLB performance with other protocols.

All of the FCTs were normalized to MDTLB for easy visualization. It is observed that MDTLB achieves less FCT than ECMP, because ECMP suffers more from hash collisions at a higher load. Moreover, MDTLB performs better at an 80% to 90% load compared to Clove and Herms for all types of flows, because MDTLB has better visibility and can react to the congestion. Furthermore, for all ranges expect 0.5–0.6, MDTLB had less FCT than most of the other protocols. MDTLB has lower a FCT than LetFlow, Drill, Presto, and CONGA for the range 0.3–0.5, because we observe that MDTLB reacts to the congestion in a more timely manner. Also, it is observed that MDTLB has achieved less than all of the other protocols in the range 0.7 until 0.9. Another observation is that the performance of MDTLB was more superior in long flows than other protocols, as we see in Figure 10b in comparison with Figure 10a,c.

Here, we focus on the FCT of short flows, long flows, and both short and long flows with the web search and data-mining workloads, which are both widely used workloads from deployed data centers, as shown in Figures 11 and 12, respectively.

Under the web-search workload

We observe that MDTLB has an average performance compared with other protocols. The reason is that the web-search considers bursty small flows. The frequent arrival of flows then creates flow gaps by breaking large flows into small flows. As shown in Figure 11, the FCT of MDTLB achieves one with the increasing load as compared to other protocols. However, excessive rerouting opportunities may negatively affect small flows without caution rerouting. As we can see in Figure 11a,c, the FCT for small flows grow dramatically as the load increases, which is because small flows are broken into several flowlets under heavy load; thus, they are heavily affected by packet reordering and congestion mismatch. In comparison, we observed that the FCT of MDTLB was lower compared with ECMP, Hermes, Clove, LetFlow, and CONGA due to being interpreted by the multi-level path characterization of MDTLB and the adaptive settings of the threshold.

Under the data-mining workload

In Figure 12, we observe that the MDTLB performance is better then Clove, Hermes, and Drill, because the data-mining workload is less bursty, and so the visibility of the network congestion becomes especially important to balance the load effectively. Also, we observe that MDTLB has achieved a higher FCT than DRB. Figure 12b shows that MDTLB outperforms ECMP, CONGA, Clove, Letflow, Presto, DRB, Hermes, and Drill; this is because MDTLB can effectively resolve the collision of large flows due to the multi-level setting aspects in it. However, MDTLB provides an FCT lower than most protocols, including DRB, when the workload increases. Furthermore, our protocol has achieved a lower FCT than Hermes, LetFlow, and Clove, because these solutions without good visibility can hardly balance the traffic.

The average throughput was evaluated for the different flows. Then, a comparison study has been made between the proposed protocol and the existing protocols (shown in Figures 13–15). It is observed that MDTLB has achieved the best throughput compared with existing protocols for data-mining flows. It is also evident that the proposed protocol is the top among three existing protocols for general flows with CONGA and Presto, while it was among the top five protocols for web flow. This is consistent with the nature of web-search flows that have small bursts flows.

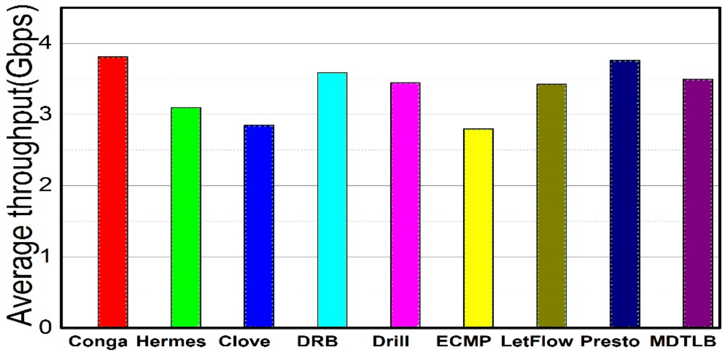


Figure 13. Average throughput comparison among the protocols for the general load.

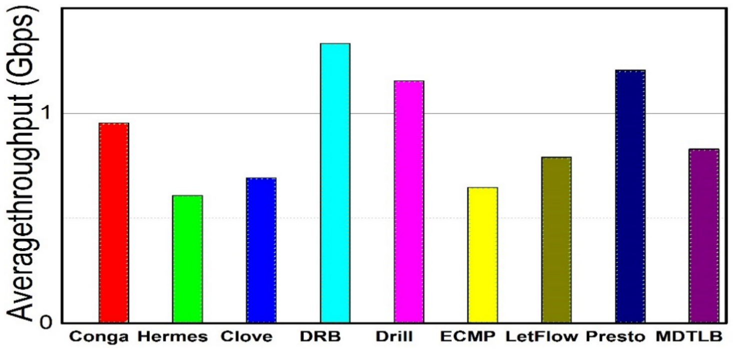


Figure 14. Average throughput comparison among the protocols for the web search load.

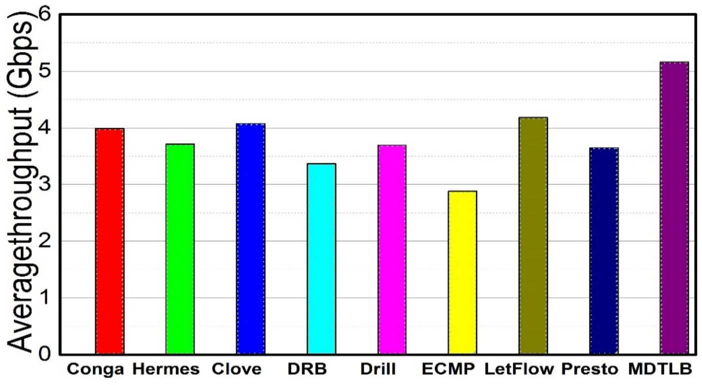


Figure 15. Average throughput comparison among the protocols for the data-mining load.

6. Related Work

The most recent and related algorithms, schemes, and protocols studies are discussed for the data center load balancing. Table 6 shows the summary of the existing scheme under the algorithm or mechanism functionality.

Researchers have proposed Presto, DRB, and RPS using per-packet per flow cell-based flow congestion load-balancing schemes [7,11]. Many researchers had proposed congestion-aware switching techniques to balance the data center loads. However, from the assessment, it is evident that most of the schemes suffer from the congestion mismatch in an asymmetric condition [12,18].

In continuation, to recover, the congestion mismatch issue advanced programmable switches that were engaged [13]. The main target was to achieve better visibility and throughput. Flowlet switching had projected balancing the traffic surrounded by parallel paths. Nevertheless, the flowlet has the limitation of reacting to the congestion situations under stable conditions that lead it to have slower convergence and suboptimal performance. A flowlet weighted round-robin based algorithm had been imposed at the end host where the path weight was estimated using piggybacked ECN signals [19].

A multipath transmission control protocol (MPTCP) [20,21] has been using as a transport protocol to routes subflows toward multiple paths. This is because it cannot change paths, which is why the MPTCP routed paths congestion mismatch issue is very low [22–24]. However, it carries some limitations on the modification of the end host networking stacks, which really gives the challenge for the multiple customers. On top of that, there are bad incast scenarios. A DRILL-based load-balancing solution had appeared, which mainly maintained the per-packet load to overcome the microbursts under heavy load conditions [25]. Moreover, it has also a limitation for the asymmetric topologies that leads to the congestion mismatch issue. Another solution, namely flowbender [26], has appeared to reroute the flows blindly whenever the congestions are detected. This technique is used to apply the random and dynamic rerouting to carry the suboptimal performance under high load conditions. In reference [27], random packet spraying (RPS) is proposed. RPS is a random based load-balancing routing algorithm, which randomly assigns packets to one of the available shortest paths to the destination. CLOVE-ECN [28] adopts per-flowlet weighted round-robin attend hosts, where the path weights are calculated based on piggybacked ECN signals, due to their limited visibility and slow reaction to congestion. The authors in [29] presented DiffFlow with the help of SDN using random packet spraying (RPS) and a new load-balancing solution to detect the long flows and forward packets. In [30], the authors presented a mechanism named flow distribution aware load balancing in order to decrease the flow completion time and obtain higher scalability. Furthermore, the DFFR algorithm [31] was proposed to balance the flows in the data center networks. It is a scalable, distributed, and adaptive algorithm designed for maximizing network resources. A data center load-balancer named Beamer is developed to guarantee a stateless mux operation [32]. Moreover, in [33], the authors introduced a load-balancing protocol that was based on sampling named Luopan in order to overcome the shortcomings of the existing protocols.

Table 6. Summary of existing scheme under uncertainties.

Chemes	Sensing Uncertainties		Reacting to Uncertainties		Advanced Hardware	Sensitivity to Parameter Settings	Evaluation Method
	Congestion	Switch Failure	Minimum Switchable Unit	Switching Method and Frequency			
Presto [11]	Oblivious	Oblivious	Flow cell (Fixed-sized unit)	Per-flow cell round robin	No	No	Real physical network
DRB [7]	Oblivious	Oblivious	Packet	Per-packet round robin	No	No	NS-3 with testbed real physical network
LetFLow [19]	Oblivious	Oblivious	Flowlet	Per-flowlet random hashing	Yes	No	NS-3
DRILL [25]	Local awareness (Switch)	Oblivious	Packet	Per-packet rerouting (according to local congestion)	Yes	No	OMNET++ Simulator
CONGA [12]	Global awareness (Switch)	Oblivious	Flowlet	Per-flowlet rerouting (according to global congestion)	Yes	No	Based on OMNET++ Simulator and real hardware testbed
HULA [13]	Global awareness (Switch)	Oblivious	Flowlet	Per-flowlet rerouting (according to global congestion)	Yes	No	NS-2 Simulator
FlowBender [26]	Global awareness (End host)	Oblivious	packet	Reactive and random rerouting (when congested)	No	No	NS-3 and real physical testbed
CLOVE-ECN [28]	Global awareness (End host)	Oblivious	Flowlet	Flowlet Per-flowlet weighted round robin (according to global congestion)	No	No	NS-2
Hermes [14]	Global awareness (End host)	Aware	Packet	Timely yet cautious rerouting (based on global congestion and failure)	No	No	NS-3
MDTLB	Global awareness (End host)	Aware	Packet	Rerouting logic (based on global congestion and failure) and path judging	No	Yes	NS-3

7. Conclusions and Future Work

This paper investigated different load-balancing mechanisms and algorithms, as well as protocols, and also discussed several of the load-balancing problems. The key consideration was to minimize the congestion mismatch problems in the proposed and designed protocol. A multi-level dynamic traffic load-balancing protocol is proposed to overcome the uncertainties (traffic dynamics, topology asymmetry, failures, and capability) issues. The proposed approach has used the multi-state path conditions and path characterizing parameters. The MDTLB simulation results outperform HERMES in terms of both FCT and throughput. Furthermore, MDTLB has achieved the best throughput compared with other protocols for data mining flows.

In the future, it will be based and tested on hardware that emulate different scenarios such as symmetry and asymmetry topologies. We can also improve the way of judging paths by implementing the dynamic thresholding method by taking T_{ECN} as the secondary input to decision-making logic.

Author Contributions: S.M., J.H., H.S., and N.K.B. conceived and designed the whole system; S.M., J.H. designed algorithm and model; S.M. and Y.A. conducted the experiment; S.M. wrote the research paper.

Funding: This work is supported by National Natural Science Foundation of China (61872387, 61572530, 61872403) and CERNET Innovation Project (Grant No. NGII20160113, NGII20170107).

Conflicts of Interest: I declare that none of the authors have a conflict of interest.

References

1. Greenberg, A.; Hamilton, J.; Maltz, D.A.; Patel, P. The cost of a cloud: Research problems in data center networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *39*, 68–73. [CrossRef]
2. Singh, A.; Korupolu, M.; Mohapatra, D. Server-storage virtualization: Integration and load balancing in data centers. In Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, Austin, TX, USA, 21 November 2008; p. 53.
3. Dillon, T.; Wu, C.; Chang, E. Cloud computing: Issues and challenges. In Proceedings of the Advanced Information Networking and Applications (AINA), Perth, Australia, 20–13 April 2010; pp. 27–33.
4. Grossman, R.L. The case for cloud computing. *IT Prof.* **2009**, *11*, 23–27. [CrossRef]
5. Chappell, D. Introducing the Windows Azure Platform. Available online: https://www.google.com.tw/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEwjEk-fjjpDgAhWaa94KHTC-CUEQFjABegQICRAC&url=http%3A%2F%2Fwww.davidchappell.com%2Fwriting%2Fwhite_papers%2FIntroducing_the_Windows_Azure_Platform%2C_v1.4--Chappell.pdf&usg=AOvVaw3BMon_jPUkscNOo26bp5nL (accessed on 16 December 2018).
6. Kannan, L.N.; Zeto, R.W.; Chen, L.; Xu, F.; Jalan, R. System and method to balance servers based on server load status. United States Patent U.S. 9,215,275, 15 December 2015.
7. Cao, J.; Xia, R.; Yang, P.; Guo, C.; Lu, G.; Yuan, L.; Zheng, Y.; Wu, H.; Xiong, Y.; Maltz, D. Per-packet load-balanced, low-latency routing for clos-based data center networks. In Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, Santa Barbara, CA, USA, 9–12 December 2013; pp. 49–60.
8. Shafiee, M.; Ghaderi, J. A simple congestion-aware algorithm for load balancing in datacenter networks. *IEEE/ACM Trans. Netw.* **2017**, *25*, 3670–3682. [CrossRef]
9. Shi, Q.; Wang, F.; Feng, D.; Xie, W. *ALB: Adaptive Load Balancing Based on Accurate Congestion Feedback for Asymmetric Topologies*; IEEE/ACM: Banff, AB, Canada, 2018.
10. Noormohammadpour, M.; Raghavendra, C.S. Datacenter Traffic Control: Understanding Techniques and Tradeoffs. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 1492–1525. [CrossRef]
11. He, K.; Rozner, E.; Agarwal, K.; Felter, W.; Carter, J.; Akella, A. Presto: Edge-based load balancing for fast datacenter networks. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, New York, NY, USA, 17–21 2015 August; Volume 45, pp. 465–478.

12. Alizadeh, M.; Edsall, T.; Dharmapurikar, S.; Vaidyanathan, R.; Chu, K.; Fingerhut, A.; Matus, F.; Pan, R.; Yadav, N.; Varghese, G. CONGA: Distributed congestion-aware load balancing for datacenters. In Proceedings of the 2014 ACM conference on SIGCOMM, New York, NY, USA, 17–22 August 2014; Volume 44, pp. 503–514.
13. Katta, N.; Hira, M.; Kim, C.; Sivaraman, A.; Rexford, J. Hula: Scalable load balancing using programmable data planes. In Proceedings of the Symposium on SDN Research ACM, Santa Clara, CA, USA, 14 March 2016; p. 10.
14. Zhang, H.; Zhang, J.; Bai, W.; Chen, K.; Chowdhury, M. Resilient datacenter load balancing in the wild. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, Los Angeles, CA, USA, 21–25 August 2017; pp. 253–266.
15. Huang, J.; Huang, Y.; Wang, J.; He, T. Adjusting Packet Size to Mitigate TCP Incast in Data Center Networks with COTS Switches. *IEEE Trans. Cloud Comput.* **2018**. [CrossRef]
16. Ruan, C.; Wang, J.; Jiang, W.; Huang, J.; Min, G.; Pan, Y. FSQCN: Fast and Simple Quantized Congestion Notification in Data Center Ethernet. *J. Netw. Comput. Appl.* **2017**, *83*, 53–62. [CrossRef]
17. Zhang, T.; Wang, J.; Huang, J.; Chen, J.; Pan, Y.; Min, G. Tuning the aggressive TCP behavior for highly concurrent HTTP connections in intra-datacenter. *IEEE/ACM Trans. Netw.* **2017**, *25*, 3808–3822. [CrossRef]
18. Wang, P.; Xu, H.; Niu, Z.; Han, D.; Xiong, Y.; Wang, P.; Xu, H.; Niu, Z.; Han, D.; Xiong, Y. Expeditus: Congestion-Aware Load Balancing in Clos Data Center Networks. *IEEE/ACM Trans. Netw. (TON)* **2017**, *25*, 3175–3188. [CrossRef]
19. Vanini, E.; Pan, R.; Alizadeh, M.; Taheri, P.; Edsall, T. Let It Flow: Resilient Asymmetric Load Balancing with Flowlet Switching. In Proceedings of the NSDI, Boston, MA, USA, 27–29 March 2017; pp. 407–420.
20. Dong, P.; Yang, W.; Tang, W.; Huang, J.; Wang, H.; Pan, Y.; Wang, J. Reducing transport latency for short flows with multipath TCP. *J. Netw. Comput. Appl.* **2018**, *108*, 20–36. [CrossRef]
21. Bonaventure, O.; Paasch, C.; Detal, G. Use Cases and Operational Experience with Multipath TCP. Available online: https://inl.info.ucl.ac.be/publications/use-cases-and-operational-experience-multipath-tcp.html (accessed on 16 December 2018).
22. Iwasaki, Y.; Ono, S.; Saruwatari, S.; Watanabe, T. Design and Implementation of OpenFlow Networks for Medical Information Systems. In Proceedings of the IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–7.
23. Frömmgen, A.; Rizk, A.; Erbshäußer, T.; Weller, M.; Koldehofe, B.; Buchmann, A.; Steinmetz, R. A programming model for application-defined multipath TCP scheduling. In Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference, Las Vegas, NV, USA, 11–15 December 2017; pp. 134–146.
24. Vo, P.L.; Le, T.A.; Tran, N.H. mFAST: A Multipath Congestion Control Protocol for High Bandwidth-Delay Connection. *Mob. Netw. Appl.* **2018**, 1–9. [CrossRef]
25. Ghorbani, S.; Yang, Z.; Godfrey, P.; Ganjali, Y.; Firoozshahian, A. DRILL: Micro load balancing for low-latency data center networks. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, Los Angeles, CA, USA, 21–25 August 2017; pp. 225–238.
26. 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.
27. Dixit, A.; Prakash, P.; Hu, Y.C.; Kompella, R.R. On the impact of packet spraying in data center networks. In Proceedings of the INFOCOM, Turin, Italy, 14–19 April 2013; pp. 2130–2138.
28. Katta, N.; Hira, M.; Ghag, A.; Kim, C.; Keslassy, I.; Rexford, J. CLOVE: How I learned to stop worrying about the core and love the edge. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks, Atlanta, GA, USA, 9–10 November 2016; pp. 155–161.
29. Carpio, F.; Engelmann, A.; Jukan, A. DiffFlow: Differentiating short and long flows for load balancing in data center networks. *arXiv*, 2016; preprint. arXiv:1604.05107.
30. Wang, S.; Zhang, J.; Huang, T.; Pan, T.; Liu, J.; Liu, Y. Flow distribution-aware load balancing for the datacenter. *Comput. Commun.* **2017**, *106*, 136–146. [CrossRef]
31. Cheung, C.M.; Leung, K.C. DFFR: A flow-based approach for distributed load balancing in Data Center Networks. *Comput. Commun.* **2018**, *116*, 1–8. [CrossRef]

32. Olteanu, V.; Agache, A.; Voinescu, A.; Raiciu, C. Stateless datacenter load-balancing with beamer. In Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Renton, WA, USA, 9–11 April 2018; Volume 18, pp. 125–139.
33. Wang, P.; Trimponias, G.; Xu, H.; Geng, Y. Luopan: Sampling-Based Load Balancing in Data Center Networks. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 133–145. [CrossRef]