


Article

EasyLB: Adaptive Load Balancing Based on Flowlet Switching for Wireless Sensor Networks

Zhiqiang Guo, Xiaodong Dong, Sheng Chen, Xiaobo Zhou *  and Keqiu Li

Tianjin Key Laboratory of Advanced Networking, School of Computer Science and Technology, Tianjin University, Tianjin 300350, China; guozhiqiang@tju.edu.cn (Z.G.); dongxiaodong@tju.edu.cn (X.D.); chensheng@tju.edu.cn (S.C.); likeqiu@gmail.com (K.L.)

* Correspondence: xiaobo.zhou@tju.edu.cn

Received: 1 August 2018; Accepted: 10 September 2018; Published: 12 September 2018



Abstract: Load balancing is effective in reducing network congestion and improving network throughput in wireless sensor networks (WSNs). Due to the fluctuation of wireless channels, traditional schemes achieving load balancing in WSNs need to maintain global or local congestion information, which turn out to be complicated to implement. In this paper, we design a flowlet switching based load balancing scheme, called EasyLB, by extending OpenFlow protocol. Flowlet switching is efficient to achieve adaptive load balancing in WSNs. Nevertheless, one tricky problem lies in determining the flowlet timeout value, δ . Setting it too small would risk reordering issue, while setting it too large would reduce flowlet opportunities. By formulating the timeout setting problem with a stationary distribution of Markov chain, we give a theoretical reference for setting an appropriate timeout value in flowlet switching based load balancing scheme. Moreover, non-equal probability path selection and multiple parallel load balancing paths are considered in timeout setting problem. Experimental results show that, by setting timeout value following the preceding theoretical reference, EasyLB is adaptive to wireless channel condition change and achieves fast convergence of load balancing after link failures.

Keywords: wireless sensor networks; flowlet switching; load balancing; Markov chain; software defined networking

1. Introduction

In wireless sensor networks (WSNs), many sensor nodes are deployed to collect various types of data from the environment, e.g., temperature, image and video. The data collected are forwarded to the sink nodes through wireless channels with or without the help of some intermediate nodes. At the sink nodes, the data are further processed to perform some specific tasks, such as fire detection, water quality monitoring and natural disaster prevention. To improve reliability and throughput, multi-path routing algorithms are widely used in WSNs [1–6], where load balancing plays a key role in reducing transmission latency and extending the lifetime of WSNs. As the wireless channels fluctuate frequently due to the inherent characteristics of wireless signals, the channel capacities of the paths change rapidly [7,8]. In the worst case, some links may even fail if deep fade happens [9]. In this scenario, how to achieve adaptive load balancing in WSNs is a crucial problem.

Many efforts focus on achieving load balancing in various scenarios. These load balancing schemes differ in operation granularity and the ability to handle asymmetry. According to load balancing granularity, load balancing schemes can be broadly divided into three categories: packet level, flow level and sub-flow level. MPTCP [10], DRB [11], Fastpass [12] and DeTail [13] are typically working at packet level. In general, the packet level schemes are able to achieve accurate control of load ratio among multi-paths, but often lead to packet reordering. ECMP [14], WCMP [15], Hedera [16]

and MicroTE [17] operate load balancing at flow level. These schemes do not generally cause packet reordering problem, but they cannot achieve accurate load balancing ratio. Flare [18], Presto [19], Conga [20] and LetFlow [21] are sub-flow level load balancing schemes that obtain a trade-off between accurate ratio control and freedom from packet reordering.

Load balancing in symmetrical network topologies such as Fat-tree [22] and Clos [23] usually splits network traffic across multiple equal paths which have the same link capacity and delay. Due to the fluctuation of wireless channels, asymmetries may occur frequently in WSNs where the capacities of parallel routing paths are no longer the same. Many existing load balancing schemes can work normally in symmetric networks. However, they will result in serious network congestion and a sharp network performance decline in WSNs when asymmetries occur, such as ECMP. Load balancing schemes can work in an asymmetric network when network congestion information is available, such as CONGA [20] and HULA [24]. However, the implementation of CONGA or HULA in WSNs is very complex because they need to obtain real-time congestion information with a centralized control of fabric and end-to-end feedback. Recently, Vanini, et al. [21] verified that adaptive load balancing can be implemented in asymmetric topology based on flowlet switching [25]. This scheme is extremely simple to implement without any explicit congestion feedback.

As shown in Figure 1, a flowlet is a sub-flow which consists of several consecutive packets from the same TCP flow. Flowlets are characterized by a timeout value, δ , which is the minimum inter-flowlet interval, i.e., packet interval within a flowlet is smaller than δ . Flowlets can be switched independently. Flowlet switching based load balancing will not cause TCP reordering if δ is set greater than the maximum delay difference between any set of parallel paths, as shown in Figure 2.

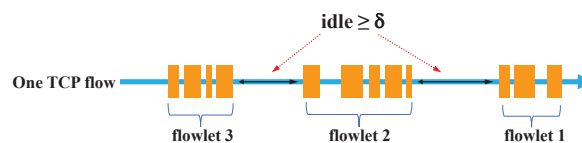


Figure 1. Flowlet is a burst of packets that is separated in time from other bursts in one TCP flow by an idle time interval that is larger than a predefined timeout value, δ .

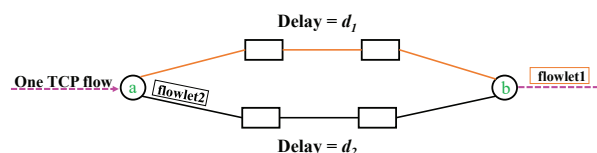


Figure 2. Flowlet switching: the packets in one flowlet follow the same path. If $\delta \geq |d_1 - d_2|$, one can randomly assign the packets in flowlet2 to one path without risking TCP packet reordering. Since packets in flowlet1 will always leave the merge point b before flowlet2.

The size of flowlet will change with the real-time network congestion. More specifically, the less congested the network is, the larger the size of flowlet is. Thus, Vanini et al. [21] found that this property of flowlet can achieve adaptive and resilient load balancing in the presence of topological asymmetry. They have implemented an extremely simple flowlet switching based load balancing scheme, called LetFlow [21]. However, a key problem lies in determining a proper timeout value. Setting the value too small will lead to heavy packet reordering issue, and setting the value too large will reduce the opportunities of flowlet generation.

However, to achieve adaptive load balancing, choosing a proper timeout value is quite difficult because of dynamic TCP traffic patterns, different capacities of multiple parallel paths, etc. Currently, the timeout is usually set empirically based on sufficient simulations and statistics. Besides, a pre-designed timeout in one network cannot always reach good performance in other scenarios.

To solve this problem, in this paper, we investigate the theoretical reference on setting an appropriate timeout value in flowlet switching based load balancing scheme, and verify the theoretical results in an OpenFlow enabled network. The contributions of this paper are summarized as follows:

- First, we design a flowlet switching based load balancing scheme for WSNs, called EasyLB, by adding one selection method for group table defined in OpenFlow [26]. We show that, by setting timeout according to the theorem, EasyLB achieves approximately optimal load balancing in WSNs and re-balances traffic quickly after link failures.
- Second, the flowlet switching process is modeled by a stationary Markov chain, with the assumption that all flows occur as the Poisson process. Based on this model, we derive a theorem that specifies the sufficient condition on timeout that ensures the system can converge to an ideal load balancing effect.
- Third, we further study the timeout setting problem when it comes to non-equal probability path selection and multiple parallel paths in flowlet switching. We conclude more generally when non-equal probability path selection is adopted in flowlet switching based load balancing scheme. Then, we give an algorithm for solving timeout setting problem in multiple parallel load balancing paths.

The paper is organized as follows. Section 2 presents the design and implementation of EasyLB. In Section 3, we describe the flow number change of parallel paths with a Markov chain model, and then we reveal the relationship between δ and the load balancing effect by the stationary distribution of Markov chain. We further study the timeout setting problem from the perspective of non-equal probability path selection and multiple parallel paths in flowlet switching. We evaluate EasyLB in different scenarios in Section 4. Section 5 concludes the paper and presents our future works.

2. EasyLB Design and Implementation

In this section, we introduce the architecture of EasyLB and give a brief explanation of the modified group table selection algorithm. EasyLB can be deployed at any merge nodes in WSNs.

The architecture of EasyLB and one example of deployment policy in WSNs is shown in Figure 3. In the WSN, the sensor nodes are reconfigurable, which is enabled by software defined networking. More specifically, the sensor nodes communicate with a common controller via OpenFlow [26]. Flowlet detection module of EasyLB is implemented in the sensor nodes while load balancing decision module resides in the controller. The controller obtains the whole network topology through the topology discovery module and periodically collects the channel state, link delay and flow information through the network monitoring module. The controller pushes the source-destination multi-path decisions into corresponding sensor nodes as group table entries.

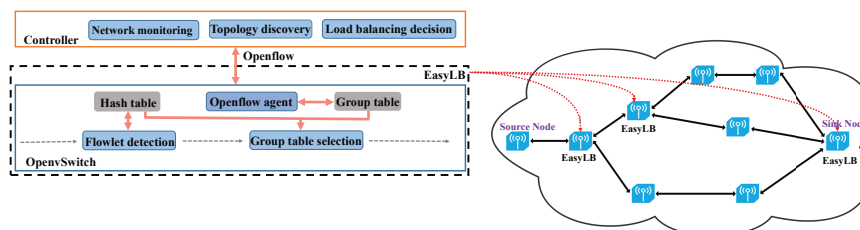


Figure 3. An overview of EasyLB and its deployment policy in wireless sensor networks.

As one representative southbound protocols in software defined networking [27], OpenFlow [26] standardizes the control signalling between control plane and data plane. Group table defined in OpenFlow protocol consists of multiple group entries and provides more advanced packet forwarding features to OpenFlow enabled switches. Each group entry contains multiple action buckets. Only

one action bucket chosen by selection method such as hash will be executed in the select type of group entry. Obviously, the group entry of type “select” is suitable for implementing multi-path forwarding. By extending the selection method, we implement EasyLB by modifying the source code of OpenvSwitch [28].

The group table selection algorithm is described in Algorithm 1; the hash tables of *last_arrival_time* and *last_output_channel* are used to record the last packet arrival time and egress channel for every flow, respectively. When the time interval between two packets belonging to one flow is greater than δ , one new flowlet is created. The flowlet switching will randomly choose one from available channels of multi-path as its egress channel. If time interval is smaller than δ , the packet will still be forwarded from the same egress channel as the previous one of the same flow. The *group_alive_buckets* function selects a normal state channel from available output channels. When channel is down, the corresponding action bucket will not be selected or executed. Meanwhile, a port down message will be sent to controller and controller will make a new multi-path decision. These mechanisms can guarantee the fast convergence of load balancing after link failures.

Algorithm 1 The group table selection algorithm.

```

1: procedure TABLESELECT
2:   last_arrival_time  $\leftarrow$  None;
3:   last_output_channel  $\leftarrow$  None;
4:   timeout  $\leftarrow$   $\delta$ ;
5:   while receive one packet do
6:     key  $\leftarrow$  hash(packet);
7:     current_time  $\leftarrow$  getTimeNow();
8:     if key not in keys of last_arrival_time then
9:       add (key, current_time) to last_arrival_time;
10:      output_channel  $\leftarrow$  random(group_alive_buckets());
11:      add (key, output_channel) to last_output_channel;
12:    else
13:      last_time  $\leftarrow$  get(key, last_arrival_time);
14:      if current_time – last_time  $\geq$  timeout then
15:        update (key, current_time) to last_arrival_time;
16:        output_channel  $\leftarrow$  random(group_alive_buckets());
17:        update (key, output_channel) to last_output_channel;
18:      else
19:        update (key, current_time) to last_arrival_time;
20:        last_channel  $\leftarrow$  get(key, last_output_channel);
21:        if last_channel corresponding bucket is not alive then
22:          output_channel  $\leftarrow$  random(group_alive_buckets());
23:          update (key, output_channel) to last_output_channel;
24:          send one port down message to controller;
25:        else
26:          output_channel  $\leftarrow$  last_channel;
27:        end if
28:      end if
29:    end if
30:    send packet to output_channel;
31:  end while
32: end procedure

```

3. Timeout Setting in EasyLB

A wrongly chosen timeout value cannot mitigate the congestion or even reduce the throughput of the whole network in flowlet switching based load balancing scheme. Generally, the value of timeout is obtained in advance through sufficient simulations and statistics. In this section, we introduce the Markov chain model to describe the flowlet switching process, and to investigate the timeout setting problem in EasyLB. We first start with the simplest case where there are two parallel paths with equal path selection, and then extend it to non-equal path selection scenario. Finally, we solve the timeout setting problem in the general multiple parallel case.

3.1. Markov Chain Model

Without loss of generality, consider that N flows transfer on two parallel paths P_1 and P_2 with bottleneck capacities C_1 and C_2 , respectively. Assume the arrivals of packets of each flow follow the Poisson process (although not all flows are subject to Poisson process in different network environments, the assumption of Poisson process can help us better understand the burstiness of TCP and carry out relevant theoretical studies, as shown in [29–32]), and all the competing flows on the same path fairly share the path’s capacity. Let \mathbb{F}_1 and \mathbb{F}_2 denote the set of flows on P_1 and P_2 , respectively. In an instant, we use γ to represent the average packet size, thus the arrival rate λ_i of flow i can be calculated as

$$\lambda_i = \begin{cases} \frac{C_1}{n_1 \gamma}, & i \in \mathbb{F}_1, \\ \frac{C_2}{n_2 \gamma}, & i \in \mathbb{F}_2, \end{cases}$$

where $n_1 = |\mathbb{F}_1|$ and $n_2 = |\mathbb{F}_2|$ denote the number of flows on P_1 and P_2 , respectively. The aggregate arrival rate λ_a on P_1 and P_2 is

$$\lambda_a = \frac{C_1 + C_2}{\gamma}.$$

Obviously, the flowlet switching process can be modeled by a Markov Chain, where the number of flows on P_1 and P_2 is the state. For state (n_1, n_2) , the next state it may transfer into is (n_1, n_2) , $(n_1 - 1, n_2 + 1)$ or $(n_1 + 1, n_2 - 1)$, which depends on the random path selection of new triggered flowlet. As Figure 4 depicts, if flow i on path P_1 triggers a new flowlet and the random selection path is P_2 , (n_1, n_2) will transfer to $(n_1 - 1, n_2 + 1)$. Similarly, if flow i on path P_2 triggers a new flowlet and the random selection path is P_1 , (n_1, n_2) will transfer to $(n_1 + 1, n_2 - 1)$. Otherwise, the state will remain unchanged.

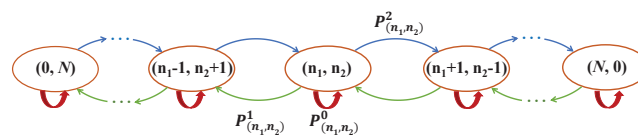


Figure 4. The model of Markov chain. In state (n_1, n_2) , there are n_1 and n_2 flows on P_1 and P_2 , respectively. The green line represents flow i on P_1 triggers a new flowlet and selects P_2 as its next path while the blue line represents flow i on P_2 triggers a new flowlet and selects P_1 as its next path. The red line represents the other cases.

Let $P^0_{(n_1, n_2)}$, $P^1_{(n_1, n_2)}$ and $P^2_{(n_1, n_2)}$ denote the transition probability from state (n_1, n_2) to (n_1, n_2) , $(n_1 - 1, n_2 + 1)$ and $(n_1 + 1, n_2 - 1)$, respectively. According to [21], the transition probability $P^1_{(n_1, n_2)}$ and $P^2_{(n_1, n_2)}$ can be calculated as

$$P^1_{(n_1, n_2)} = \frac{1}{2} \sum_{i \in \mathbb{F}_1} (f(\lambda_i) + g(\lambda_i)), \quad n_1 \in [1, N],$$

and

$$P^2_{(n_1, n_2)} = \frac{1}{2} \sum_{i \in \mathbb{F}_2} (f(\lambda_i) + g(\lambda_i)), n_1 \in [0, N - 1],$$

where $f(\lambda_i) = \frac{\lambda_i}{\lambda_a - \lambda_i} (e^{-\lambda_i \delta} - e^{-\lambda_a \delta})$ and $g(\lambda_i) = \frac{\lambda_i}{\lambda_a} e^{-\lambda_a \delta}$.

Since $\lambda_i \ll \lambda_a$, $P^0_{(n_1, n_2)}$, $P^1_{(n_1, n_2)}$ and $P^2_{(n_1, n_2)}$ can be approximated as

$$\begin{aligned} P^1_{(n_1, n_2)} &\approx \frac{C_1}{2(C_1 + C_2)} e^{-\frac{C_1 \delta}{n_1 \gamma}}, n_1 \in [1, N], \\ P^2_{(n_1, n_2)} &\approx \frac{C_2}{2(C_1 + C_2)} e^{-\frac{C_2 \delta}{n_2 \gamma}}, n_1 \in [0, N - 1], \\ P^0_{(n_1, n_2)} &= 1 - P^1_{(n_1, n_2)} - P^2_{(n_1, n_2)}, n_1 \in [1, N - 1]. \end{aligned}$$

Note that the transition probabilities of $P^0_{(0, N)}$ and $P^0_{(N, 0)}$ are $1 - P^2_{(0, N)}$ and $1 - P^1_{(N, 0)}$, respectively.

The transition probability between (n_1, n_2) and any other state besides (n_1, n_2) , $(n_1 - 1, n_2 + 1)$ and $(n_1 + 1, n_2 - 1)$ is 0. In summary, the transition probability matrix \mathbf{P} of this Markov chain can be written as

$$\mathbf{P} = \begin{bmatrix} P^0_{(0, N)} & P^2_{(0, N)} & & & & \\ P^1_{(1, N-1)} & P^0_{(1, N-1)} & P^2_{(1, N-1)} & & & \\ & \ddots & \ddots & \ddots & & \\ & & P^1_{(N-1, 1)} & P^0_{(N-1, 1)} & P^2_{(N-1, 1)} & \\ & & & P^1_{(N, 0)} & P^0_{(N, 0)} & \end{bmatrix}$$

which is a typical tridiagonal matrix.

3.2. Formalization of Timeout Setting Problem

In this subsection, we derive the sufficient condition on timeout that ensures the flowlet switching based load balancing scheme to achieve ideal load balancing effect.

Definition 1. In the flowlet switching based load balance scheme considered in the previous section, if $\frac{n_1}{n_2} = \frac{C_1}{C_2}$, the ideal load balancing effect is achieved.

We define μ as the number of flows mapping to P_1 when the ideal load balancing effect is achieved. According to Definition 1, $\mu = \left\lfloor \frac{C_1}{C_1 + C_2} \times N \right\rfloor$. In this case, the corresponding state of the Markov chain is $(\mu, N - \mu)$, we call it the ideal state.

Theorem 1. In flowlet switching based load balancing scheme, if N flows transmit over two parallel paths with capacities C_1 and C_2 , respectively, the sufficient condition on timeout δ to achieve ideal load balancing effect is $\delta > \delta_{\min}$, where

$$\delta_{\min} = \begin{cases} 0, & \text{if } C_1 = C_2, \\ \frac{\mu(N-\mu+1)\gamma \times \ln(\frac{C_1}{C_2})}{C_1(N-\mu+1) - C_2\mu}, & \text{if } C_1 > C_2, \\ \frac{(N-\mu)(\mu+1)\gamma \times \ln(\frac{C_1}{C_2})}{C_1(N-\mu) - C_2(\mu+1)}, & \text{if } C_1 < C_2. \end{cases} \tag{1}$$

proof of Theorem 1. The number of states in the preceding Markov chain is restricted and all states are accessible, thus this Markov chain is irreducible and all states are recurrent. Additionally, any state can access itself again through one step, so this Markov chain has stationary distribution. Let $\vec{\pi} = [\pi_0, \pi_1, \dots, \pi_m, \dots, \pi_N]$ represent the stationary distribution of this Markov chain, where π_m denotes the stationary probability of state $(m, N - m)$. We can get the stationary distribution by solving

$$\vec{\pi} \mathbf{P} = \vec{\pi}$$

where $\sum_{m=0}^N \pi_m = 1$. After some mathematical manipulations, we finally get:

$$\frac{\pi_m}{\pi_{m+1}} = \frac{P_{(m+1, N-m-1)}^1}{P_{(m, N-m)}^2} \quad 0 \leq m \leq N-1. \quad (2)$$

The relationship of stationary probability between *ideal state* and that of any other state can be deduced from Equation (2) as:

$$\begin{aligned} \frac{\pi_\mu}{\pi_{\mu+k}} &= \frac{\pi_\mu}{\pi_{\mu+1}} \times \frac{\pi_{\mu+1}}{\pi_{\mu+2}} \times \dots \times \frac{\pi_{\mu+k-1}}{\pi_{\mu+k}} \\ &= \left(\frac{C_1}{C_2}\right)^k e^{-\delta \sum_{j=1}^k \left(\frac{C_1}{(\mu+j)\gamma} - \frac{C_2}{(N-\mu-j+1)\gamma}\right)}, \quad k \in [1, N-\mu], \end{aligned} \quad (3)$$

$$\begin{aligned} \frac{\pi_{\mu-k}}{\pi_\mu} &= \frac{\pi_{\mu-1}}{\pi_\mu} \times \frac{\pi_{\mu-2}}{\pi_{\mu-1}} \times \dots \times \frac{\pi_{\mu-k}}{\pi_{\mu-k+1}} \\ &= \left(\frac{C_1}{C_2}\right)^k e^{-\delta \sum_{j=1}^k \left(\frac{C_1}{(\mu-j+1)\gamma} - \frac{C_2}{(N-\mu+j)\gamma}\right)}, \quad k \in [1, \mu]. \end{aligned} \quad (4)$$

From the perspective of stationary distribution, when it comes to ideal load balancing effect, the stationary probability of *ideal state* is greater than any other one, i.e.,

$$\frac{\pi_\mu}{\pi_{\mu+k}} > 1, \quad \forall k \in [1, N-\mu], \quad (5)$$

$$\frac{\pi_{\mu-k}}{\pi_\mu} < 1, \quad \forall k \in [1, \mu]. \quad (6)$$

By solving Equations (5) and (6), we can obtain the results specified in Equation (1). For more detailed derivation, please refer to Appendix A. \square

When the capacity of P_1 is greater than that of P_2 , Equation (3) is always larger than 1. That is, when P_1 has more flows than *ideal state*, flows themselves incline to transfer to P_2 . Contrarily, when the capacity of P_1 is smaller than that of P_2 , Equation (4) is always smaller than 1. This indicates that, when P_1 has fewer flows than *ideal state*, flows tend to transfer to P_1 . Note that, when paths are symmetric, the timeout has no effect on the final load balancing state.

Assuming that the delay difference between P_1 and P_2 is d , δ not only has to meet the constraints in Theorem 1, but also has to be greater than d to avoid packet reordering. However, setting δ too large also lowers the possibility of flowlet generation. An extreme case is setting δ at infinity, which will result in a flow-level load balancing scheme. An upper bound on δ is left as a future study.

3.3. Timeout Setting in Non-Equal Probability Path Selection

It is shown in [21] that, with equal probability path selection, flowlet switching based load balancing scheme can achieve adaptive load balancing on the premise of appropriate selection of δ , which is the magic of this load balancing technology. Considering a more general case where the paths are selected with non-equal probabilities, it is worth investigating whether the flowlet switching based load balancing scheme can still maintain its adaptive load balancing capability. In that case, how should δ be set? In practice, the paths are usually selected with non-equal probabilities; let us take the following two scenarios as an example. The first scenario is the switch queue's length on path P_1 is greater than that on path P_2 . To reduce the packet loss rate and improve overall performance of the network, we should choose path P_1 as new routing path at a higher probability for newly triggered flowlet. The second scenario is when the bandwidth of path P_1 is greater than that of P_2 ; for the purpose of speeding up the convergence rate of load balancing, we choose P_1 as routing path for newly triggered flowlet at a higher probability. In this section, we study the timeout setting problem when non-equal probability path selection is applied to flowlet switching based load balancing scheme.

Denote the probability that P_1 and P_2 are selected as the routing path for newly triggered flowlet as p and q , respectively. p and q meet the following conditions (to accelerate the convergence process of load balancing, p and q are usually set proportional to the bandwidth of the corresponding paths, i.e., $p = \frac{C_1}{C_1+C_2}, q = \frac{C_2}{C_1+C_2}$), where

$$\begin{cases} p > 0, \\ q > 0, \\ p + q = 1, \\ p \neq q. \end{cases}$$

It is easy to obtain the transition probability $P_{(n_1, n_2)}^1$ as

$$P_{(n_1, n_2)}^1 = q \times \sum_{i \in \mathbb{F}_1} (f(\lambda_i) + g(\lambda_i)), n_1 \in [1, N].$$

Similarly, the transition probability $P_{(n_1, n_2)}^2$ is

$$P_{(n_1, n_2)}^2 = p \times \sum_{i \in \mathbb{F}_2} (f(\lambda_i) + g(\lambda_i)), n_1 \in [0, N - 1].$$

According to Equation (2), we can get the relationship of stationary probability between ideal state and that of any other state is

$$\begin{aligned} \frac{\pi_{\mu}}{\pi_{\mu+k}} &= \frac{\pi_{\mu}}{\pi_{\mu+1}} \times \frac{\pi_{\mu+1}}{\pi_{\mu+2}} \times \dots \times \frac{\pi_{\mu+k-1}}{\pi_{\mu+k}} \\ &= \left(\frac{C_1}{C_2} \times \frac{q}{p}\right)^k e^{-\delta \sum_{j=1}^k \left(\frac{C_1}{(\mu+j)\gamma} - \frac{C_2}{(N-\mu-j+1)\gamma}\right)}, k \in [1, N - \mu], \\ \frac{\pi_{\mu-k}}{\pi_{\mu}} &= \frac{\pi_{\mu-1}}{\pi_{\mu}} \times \frac{\pi_{\mu-2}}{\pi_{\mu-1}} \times \dots \times \frac{\pi_{\mu-k}}{\pi_{\mu-k+1}} \\ &= \left(\frac{C_1}{C_2} \times \frac{q}{p}\right)^k e^{-\delta \sum_{j=1}^k \left(\frac{C_1}{(\mu-j+1)\gamma} - \frac{C_2}{(N-\mu+j)\gamma}\right)}, k \in [1, \mu]. \end{aligned}$$

Following Equations (5) and (6), we have

$$\delta_{\min} = \begin{cases} 0, & \text{if } qC_1 = pC_2, \\ \frac{\mu(N-\mu+1)\gamma \times \ln\left(\frac{C_1}{C_2} \times \frac{q}{p}\right)}{C_1(N-\mu+1) - C_2\mu}, & \text{if } qC_1 > pC_2, \\ \frac{(N-\mu)(\mu+1)\gamma \times \ln\left(\frac{C_1}{C_2} \times \frac{q}{p}\right)}{C_1(N-\mu) - C_2(\mu+1)}, & \text{if } qC_1 < pC_2. \end{cases} \tag{7}$$

As long as $\delta > \delta_{\min}$, adaptive load balancing can be achieved even with non-equal probability path selection. The derivation process is similar to Appendix A It can be seen that Theorem 1 is actually a special case with $p = q = 0.5$.

3.4. Timeout Setting in Multiple Parallel Load Balancing Paths

In this subsection, we study the problem of how to set δ to achieve adaptive load balancing when the load balancing parallel paths are multiple. When there are multiple parallel load balancing paths, each newly triggered flowlet may choose any of the paths as new routing path. As the number of multiple parallel paths increases, the Markov chain model encounters the state explosion problem and hence the computation of the transition probability matrix and the stationary distribution will become intractable. Alternatively, we leverage the results shown in Equation (7) to derive the threshold on δ .

Assume there are M parallel paths P_1, P_2, \dots, P_M with capacities C_1, C_2, \dots, C_M , respectively. We divide all paths into two logical paths \mathbb{P}_1 and \mathbb{P}_2 , where \mathbb{P}_1 contains P_1 only and \mathbb{P}_2 contains all the other paths. The bandwidth of \mathbb{P}_1 and \mathbb{P}_2 are $C_1 = C_1$ and $C_2 = \sum_{k=2}^M C_k$, respectively. With random path

selection of the M paths, the probability that \mathbb{P}_1 or \mathbb{P}_2 is chosen as the routing path for newly triggered flowlet is $p = \frac{1}{M}$ and $q = \frac{M-1}{M}$, respectively. The number of flows mapping to \mathbb{P}_1 can be calculated as $\mu = \left\lfloor \frac{C_1}{C_1+C_2} \times N \right\rfloor$ when ideal load balancing effect is achieved. According to Equation (7), we can obtain δ_{\min}^1 that specifies the lower bound on δ when ideal load balance effect is achieved between \mathbb{P}_1 and \mathbb{P}_2 . After obtaining δ_{\min}^1 , we remove P_1 from the physical paths resulting in a new problem of how to achieve load balance between the remaining $M-1$ paths P_2, P_3, \dots, P_M . Similarly, we can obtain δ_{\min}^2 by regarding P_2 as one logical path and all the other $M-2$ paths as another logical path. This procedure can be processed recursively until there are only two paths P_{M-1} and P_M left which yields δ_{\min}^{M-1} . When ideal load balance effect is achieved between the M parallel paths, δ must be set as $\delta > \delta_{\min}$, where

$$\delta_{\min} = \max\{\delta_{\min}^k\}, k \in [1, M-1].$$

The detailed algorithm is described in Algorithm 2. The time complexity of the algorithm is $\mathcal{O}(M)$,

Algorithm 2 An iterative algorithm for solving timeout setting problem with multiple parallel paths.

Input:

- One physical path set, $P(P_1, P_2, \dots, P_i, \dots, P_M) (2 \leq i \leq M)$;
- One logic path set that contains physical path in $P, \mathbb{P}_k (k = 1 \text{ or } 2)$;
- The bandwidth of physical path P_i, C_i ;
- The bandwidth of $\mathbb{P}_k, C_k (k = 1 \text{ or } 2)$;
- The total flow number, N ;
- The number of flows mapping to \mathbb{P}_1 when ideal load balancing effect is achieved, μ ;
- The probabilities of \mathbb{P}_1 and \mathbb{P}_2 is chosen as the routing path for newly triggered flowlet, p and q ;
- The m -th minimum value which must be less than $\delta, \delta_m (1 \leq m \leq M-1)$;

Output:

- The maximum value of δ_m, δ_{\min} ;
 - 1: **for** m from 1 to $M-1$ **do**
 - 2: $P_p \leftarrow$ pick the first physical path in P ;
 - 3: $\mathbb{P}_1 \leftarrow (P_p)$ and $\mathbb{P}_2 \leftarrow P - \mathbb{P}_1$;
 - 4: $C_1 \leftarrow \sum_{P_i \in \mathbb{P}_1} C_i$ and $C_2 \leftarrow \sum_{P_i \in \mathbb{P}_2} C_i$;
 - 5: $\mu \leftarrow \left\lfloor \frac{C_1}{C_1+C_2} \times N \right\rfloor$;
 - 6: $p \leftarrow \frac{1}{M+1-m}$ and $q \leftarrow \frac{M-m}{M+1-m}$;
 - 7: According to (7), we get the δ_m ;
 - 8: $N \leftarrow N - \mu$;
 - 9: Remove P_p from physical path set P ;
 - 10: **end for**
 - 11: $\delta_{\min} \leftarrow \max(\delta_m) (1 \leq m \leq M-1)$;
 - 12: **return** δ_{\min} ;
-

where M is the number of total parallel load balancing paths.

4. Performance Evaluation

In this section, we evaluate the performance of EasyLB both in symmetric topology and asymmetric topology with random path selection. Further, we evaluate EasyLB under non-equal path selection and multiple parallel paths in flowlet switching. Experimental results show that EasyLB achieves relatively ideal load balancing effect. Meanwhile, without maintaining explicit congestion

information, EasyLB has the ability to handle asymmetry in the topology and achieves fast convergence of load balancing after link failures.

4.1. Asymmetric Topology with Random Path Selection

The testbed topology of wireless sensor networks is shown in Figure 5. The testbed is based on SDN-WISE [33] which is a prototype system of SD-WSNs and uses OMNeT++ [34] simulator. We break down P_3 in the simulation of asymmetric topology and symmetric topology. The capacity of P_1 and P_2 are set to 20 and 10 Kbps, respectively. The delay difference between P_1 and P_2 is 0.01 s. The forwarding queues of the sensor nodes are running in a round-robin [35] fashion in our simulations to enhance the fairness of links. Ten long-lived TCP flows are generated that transmit from node C_1 to C_2 . The randomly generated requests are distributed to P_1 and P_2 . The average packet size of each flow is 80 Bytes. We run a basic flowlet switching process whose path selection is random, as described in Algorithm 1. We vary timeout value from 0.2 s to 8 s and get the convergent traffic load ratio of P_1 to P_2 as shown in Figure 6a. According to Equation (1), δ_{min} equals to 0.64 s. When $\delta > \delta_{min}$, such as 1 s or 1.2 s, the system will achieve the ideal load balancing effect. However, if the value of timeout is set too large, such as bigger than 2 s, the granularity of flowlet switching based load balancing scheme is approximate to flow level and the traffic load ratio of P_1 to P_2 will be much more random. Similarly, if timeout is set too small, such as 0.2 s, it turns out to be packet-level load balancing and the traffic distributed onto this two paths is roughly the same.

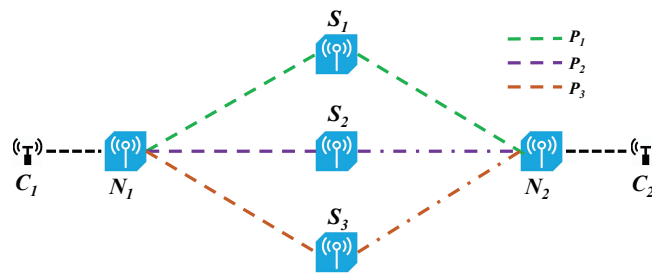


Figure 5. The testbed topology of wireless sensor networks.

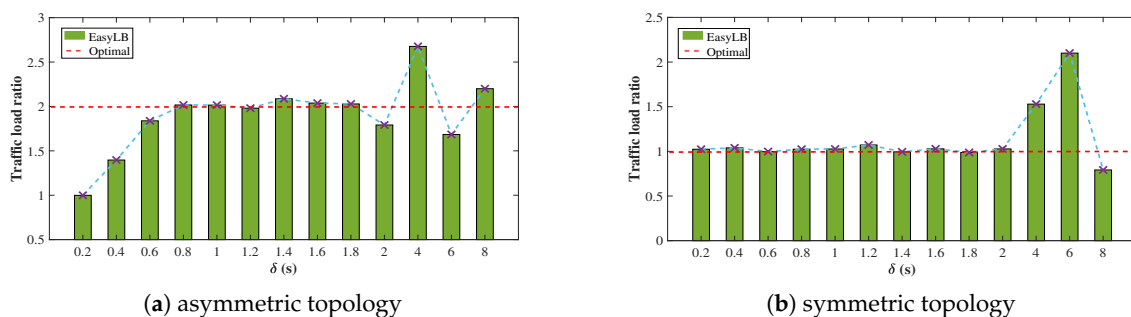


Figure 6. The convergent traffic load ratio of P_1 to P_2 under different timeout values.

In Figure 7, we set timeout to different values to show the real time traffic load ratio of P_1 to P_2 . As shown in Figure 7a,b, the traffic load ratio fluctuates in both symmetric and asymmetric cases. As more flowlets are assigned to one path, congestion is more likely to occur on this path which results in the flowlet size reduction. Flows will transfer from a more congested path to a less congested one, and finally the size of flowlet on this two paths is balanced.

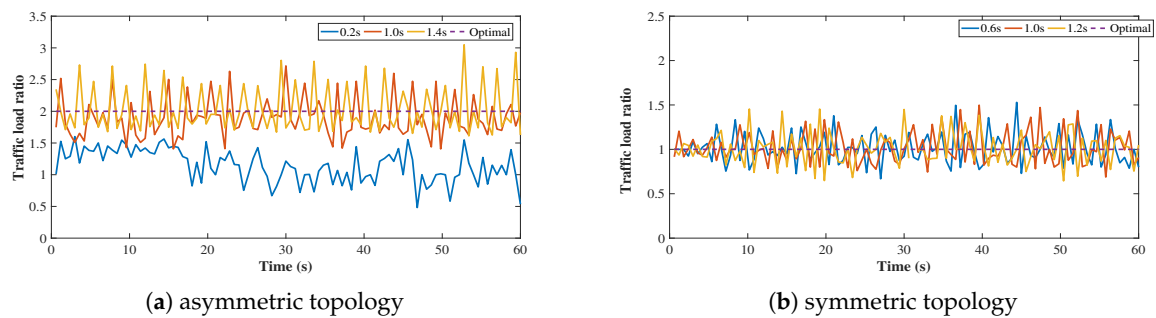


Figure 7. The traffic load ratio change of P_1 to P_2 under different timeout values.

4.2. Symmetric Topology with Random Path Selection

We also evaluate the performance of EasyLB in symmetric topology, where the capacity of P_1 and P_2 are both set to 10 Kbps. The convergent traffic load ratio of P_1 to P_2 is shown in Figure 6b. According to Equation (1), δ_{\min} equals to 0 s. As depicted in Figure 6b, the traffic load ratio is approximately 1, which is the ideal load balancing effect in symmetric topology as long as the value of δ is not too large.

The load ratio change of P_1 to P_2 in symmetric topology is shown in Figure 7b. We can find that EasyLB converges to ideal load balance effect. We know that, when the number of flows in one path increases, this path becomes more congested and more packets are likely to be dropped. Packet loss in TCP brings about TCP timeout and a new flowlet will be generated, which forces flows switch to another path. When the system eventually reaches steady state, the congestion degree is almost the same.

4.3. Non-Equal Probability Path Selection

To evaluate EasyLB with non-equal probability path selection, we set the path selection probabilities of P_1 and P_2 to $\frac{1}{3}$ and $\frac{2}{3}$, respectively. The bandwidth of P_1 and P_2 is set to 20 and 10 Kbps, respectively. According to Equation (7), δ_{\min} equals to 1.28 s. As shown in Figure 8, as long as δ is larger than δ_{\min} , the traffic load ratio of P_1 to P_2 approximately approaches 2, which is the ideal load balancing effect. Note that, as δ is sufficiently large, it acts more like flow-level load balancing and hence the traffic load ratio of P_1 to P_2 has difficulty converging to 2, as shown in Figure 8. An interesting finding is that, although the path selection probabilities of P_1 and P_2 are inversely proportional to their bandwidth, EasyLB maintains its adaptive load balancing capability as long as the timeout is appropriately set.

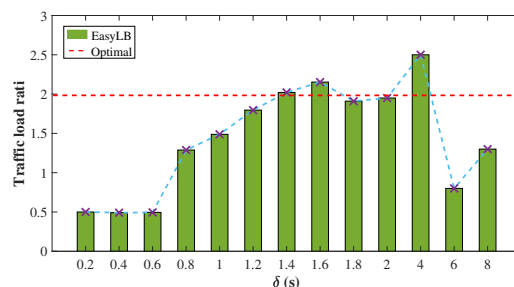


Figure 8. The convergent traffic load ratio of P_1 to P_2 with non-equal path selection under different timeout values.

4.4. Multiple Parallel Paths

In this subsection, we evaluate the performance of EasyLB in the presence of multiple paths. As shown in Figure 5, the client applications on sensor node C_1 randomly generate requests to server applications on C_2 . The capacity of the three paths P_1 , P_2 , and P_3 are set to 6, 6 and 18 Kbps, respectively. We randomly assign new triggered flowlet to the three paths, thus the path selection probability of each path is $\frac{1}{3}$. According to Algorithm 2, δ_{\min} equals 0.88 s. As shown in Figure 9, when δ is set smaller than δ_{\min} , EasyLB acts more like a packet-level load balancing scheme, therefore traffic loads of the three paths are almost the same. When δ is set larger than δ_{\min} , EasyLB achieves the ideal load balancing effect, in which the traffic distributed to each path is proportional to its bandwidth. However, as noted above, when δ is too large, EasyLB evolves to flow level load balance scheme, thus the final traffic load ratio of the three paths is much more random.

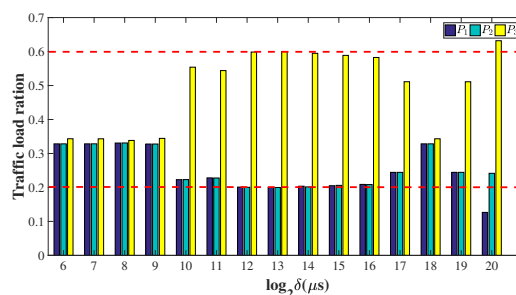


Figure 9. The convergent traffic load ratio of P_1 , P_2 and P_3 under different timeout values.

4.5. React to Link Failure

The capacity of three paths P_1 , P_2 , and P_3 are set to 6, 6 and 18 Kbps, respectively. We set δ to 1.2 s and 1.4 s. At first, the traffic is distributed over the three paths, and around 20% traffic is allocated to P_2 . At $t = 30$ s, we manually break the link (N_1, S_1) . As is shown in Figure 10, EasyLB reacts fast to this link failure event and around 25% of traffic is allocated to P_2 after the re-balance. This is because, once link failure happens in multi-paths case, the SDN controller can perceive it quickly and pushes the new multi-path decision into switches as a new group entry.

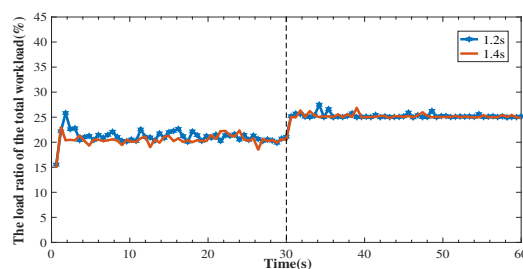


Figure 10. The traffic load change of P_2 under timeout value 1.2 s and 1.4 s. At $t = 30$ s, a link down event occurs on P_1 .

5. Conclusions and Future Work

We have studied the relationship between the timeout value in flowlet switching and the final state of load balancing with a stationary Markov chain. We further derived a theorem that specifies the sufficient condition on timeout to achieve ideal load balancing effect, which gives a comprehensive recommendation on setting timeout in different network environments. We also implemented flowlet switching based load balancing scheme called EasyLB in software defined networking by extending OpenFlow protocol. Experimental results show that, by setting the timeout following the theorem,

EasyLB has adaptive load balancing ability both in symmetric topology and in asymmetric topology. The number of active flows was assumed to be static. However, in highly dynamic networks [36,37], such as vehicle-to-vehicle (V2V) networks, the number of flows changes rapidly. Moreover, the flow priority may affect the timeout value setting. How to deal with the dynamic arrival and departure of flows, as well as take the flow priority into account in timeout value setting, is left as future studies.

Author Contributions: Conceptualization, Z.G.; Methodology, Z.G.; Software, Z.G.; Validation, K.L.; Formal Analysis, Z.G. and X.Z.; Investigation, X.D.; Resources, X.Z.; Data Curation, Z.G.; Writing-Original Draft Preparation, Z.G.; Writing-Review and Editing, X.D., S.C. and X.Z.; Visualization, S.C.; Supervision, X.Z. and K.L.; Project Administration, X.Z. and K.L.; Funding Acquisition, X.Z. and K.L.

Funding: This work was supported in part by the National Key Research and Development Program of China (No. 2016YFB1000205); in part by the State Key Program of National Natural Science of China (Grant No. 61432002); in part by the National Natural Science Foundation of China - Guangdong Joint Fund (Grant No. U1701263); in part by the National Natural Science Foundation of China (Grant Nos. 61702365, 61672379, 61772112); in part by the Natural Science Foundation of Tianjin (Grant Nos. 17JQNJJC00700, 17JCYBJC15500), and also in part by the Special Program of Artificial Intelligence of Tianjin Municipal Science and Technology Commission (Grant No. 17ZXRGX00150).

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

Appendix A The Derivation Process of Theorem 1

Taking the natural logarithm of both sides of Equations (5) and (6), we have

$$\begin{aligned} \ln \frac{\pi_{\mu}}{\pi_{\mu+k}} &= -\delta \sum_{j=1}^k \left(\frac{C_1}{(\mu+j)\gamma} - \frac{C_2}{(N-\mu-j+1)\gamma} \right) \\ &\quad + k \times \ln \frac{C_1}{C_2} > 0, \quad \forall k \in [1, N-\mu], \end{aligned} \quad (A1)$$

$$\begin{aligned} \ln \frac{\pi_{\mu-k}}{\pi_{\mu}} &= -\delta \sum_{j=1}^k \left(\frac{C_1}{(\mu-j+1)\gamma} - \frac{C_2}{(N-\mu+j)\gamma} \right) \\ &\quad + k \times \ln \frac{C_1}{C_2} < 0, \quad \forall k \in [1, \mu]. \end{aligned} \quad (A2)$$

We can get $\frac{C_1}{(\mu+j)\gamma} - \frac{C_2}{(N-\mu-j+1)\gamma}$ is always less than zero for $j \in [1, N-\mu]$ and $\frac{C_1}{(\mu-j+1)\gamma} - \frac{C_2}{(N-\mu+j)\gamma}$ is always larger than zero for $j \in [1, \mu]$.

Thus, it can be derived from Equations (8) and (9) that δ should satisfy the following inequality as $\delta > \delta_{\min} = \max\{\delta_{\min 1} = \max_{1 \leq k \leq N-\mu} (f(k)), \delta_{\min 2} = \max_{1 \leq k \leq \mu} (g(k))\}$ where

$$f(k) = \frac{k \times \ln \frac{C_1}{C_2}}{\sum_{j=1}^k \left(\frac{C_1}{(\mu+j)\gamma} - \frac{C_2}{(N-\mu-j+1)\gamma} \right)}$$

and

$$g(k) = \frac{k \times \ln \frac{C_1}{C_2}}{\sum_{j=1}^k \left(\frac{C_1}{(\mu-j+1)\gamma} - \frac{C_2}{(N-\mu+j)\gamma} \right)}.$$

If $C_1 = C_2$, $\ln \frac{C_1}{C_2} = 0$ and $\delta_{\min 1} = \delta_{\min 2} = 0$, then $\delta_{\min} = 0$. If $C_1 > C_2$, $\ln \frac{C_1}{C_2} > 0$ and $\delta_{\min 1} < 0$, $\delta_{\min 2} > 0$, then $\delta_{\min} = \delta_{\min 2}$. Similarly, if $C_1 < C_2$, $\ln \frac{C_1}{C_2} < 0$ and $\delta_{\min 1} > 0$, $\delta_{\min 2} < 0$, then $\delta_{\min} = \delta_{\min 1}$. Further, $f(k)$ and $g(k)$ are both decreasing, thus $\delta_{\min 1}$ and $\delta_{\min 2}$ get their maximum values when $k = 1$. In conclusion, we can get the final results in Equation (1).

References

1. Radi, M.; Dezfouli, B.; Bakar, K.A.; Lee, M.J.S. Multipath routing in wireless sensor networks: Survey and research challenges. *Sensors* **2012**, *12*, 650–685. [[CrossRef](#)] [[PubMed](#)]

2. Ganesan, D.; Govindan, R.; Shenker, S.; Estrin, D. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **2001**, *5*, 11–25. [[CrossRef](#)]
3. Pham, Q.V.; Hwang, W.J. Network utility maximization in multipath lossy wireless networks. *Int. J. Commun. Syst.* **2017**, *30*, 1–18. [[CrossRef](#)]
4. Pham, Q.-V.; Hwang, W.J. Network utility maximization-based congestion control over wireless networks: A survey and potential directives. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1173–1200. [[CrossRef](#)]
5. Liu, X.; Xie, X.; Zhao, X.; Li, K.; Liu, A.X.; Guo, S.; Wu, J. Fast identification of blocked rfid tags. *IEEE Trans. Mob. Comput.* **2018**, *17*, 2041–2054. [[CrossRef](#)]
6. Liu, X.; Xiao, B.; Li, K.; Liu, A.X.; Wu, J.; Xie, X.; Qi, H. RFID estimation with blocker tags. *IEEE/ACM Trans. Netw.* **2017**, *25*, 224–237. [[CrossRef](#)]
7. Liu, X.; Xie, X.; Li, K.; Xiao, B.; Wu, J.; Qi, H.; Lu, D. Fast tracking the population of key tags in large-scale anonymous rfid systems. *IEEE/ACM Trans. Netw.* **2017**, *25*, 278–291. [[CrossRef](#)]
8. Liu, X.; Li, K.; Guo, S.; Liu, A.X.; Li, P.; Wang, K.; Wu, J.; Liu, X.; Li, K.; Guo, S. Top-k queries for categorized rfid systems. *IEEE/ACM Trans. Networking* **2017**, *25*, 2587–2600. [[CrossRef](#)]
9. He, J.; Tervo, V.; Zhou, X.; He, X.; Qian, S.; Cheng, M.; Juntti, M.; Matsumoto, T. A tutorial on lossy forwarding cooperative relaying. *IEEE Commun. Surv. Tutor.* **2018**. [[CrossRef](#)]
10. Raiciu, C.; Barre, S.; Plunke, C.; Greenhalgh, A.; Wischik, D.; Handley, M. Improving datacenter performance and robustness with multipath tcp. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 266–277. [[CrossRef](#)]
11. 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.
12. Perry, J.; Ousterhout, A.; Balakrishnan, H.; Shah, D.; Fugal, H. Fastpass: A centralized zero-queue datacenter network. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *44*, 307–318. [[CrossRef](#)]
13. Zats, D.; Das, T.; Mohan, P.; Borthakur, D.; Katz, R. DeTail: Reducing the flow completion time tail in datacenter networks. In Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Helsinki, Finland, 13–17 August 2012; pp. 139–150.
14. Hopps, C. Analysis of An Equal-Cost Multi-Path Algorithm. *Rfc* **2000**, *109*, S265.
15. Zhou, J.; Tewari, M.; Zhu, M.; Kabbani, A.; Poutievski, L.; Singh, A.; Vahdat, A. WCMP: Weighted cost multipathing for improved fairness in data centers. In Proceedings of the Ninth European Conference on Computer Systems, Amsterdam, The Netherlands, 14–16 April 2014.
16. Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. Hedera: Dynamic flow scheduling for data center networks. In Proceedings of the NSDI'10 Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, San Jose, CA, USA, 28–30 April 2010.
17. 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.
18. Kandula, S.; Katabi, D.; Sinha, S.; Berger, A. Dynamic load balancing without packet reordering. *ACM SIGCOMM Comput. Commun. Rev.* **2007**, *37*, 51–62. [[CrossRef](#)]
19. 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](#)]
20. 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, Chicago, IN, USA, 17–22 August 2014.
21. Vanini, E.; Pan, R.; Alizadeh, M.; Taheri, P.; Edsall, T. Let It Flow: Resilient asymmetric load balancing with flowlet switching. In Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation, Boston, MA, USA, 27–29 March 2017; pp 407–420.
22. Al-Fares, M.; Loukissas, A.; Vahdat, A. A scalable, commodity data center network architecture. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 63–74. [[CrossRef](#)]
23. Greenberg, A.; Hamilton, J.R.; Jain, N.; Kandula, S.; Kim, C.; Lahiri, P.; Maltz, D.A.; Patel, P.; Sengupta, S. VL2: A scalable and flexible data center network. In Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, Barcelona, Spain, 16–21 August 2009; pp. 51–62.

24. 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, Santa Clara, CA, USA, 14–15 March 2016.
25. Sinha, S.; Kandula, S.; Katabi, D. Harnessing tcp's burstiness with flowlet switching. In Proceedings of the Third Workshop on Hot Topics in Networks HotNets-III, San Diego, CA, USA, 15–16 November 2004.
26. OpenFlow Switch Specification Version 1.3.1. Available online: <https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.3.1.pdf> (accessed on 9 September 2018).
27. Foundation, O. Software-defined networking: The new norm for networks. *ONF White Pap.* **2012**, *2*, 2–6.
28. Production Quality, Multilayer Open Virtual Switch. Available online: <http://www.openvswitch.org> (accessed on 9 September 2018).
29. Chandrasekaran, S.S. Understanding Traffic Characteristics in a Server to Server Data Center Network. Master's Thesis, Rochester Institute of Technology, Rochester, NY, USA, 2017.
30. Bai, W.; Chen, K.; Wang, H.; Chen, L.; Han, D.; Tian, C. Information-agnostic flow scheduling for commodity data centers. In Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, Oakland, CA, USA, 4–6 May 2015; pp. 455–468.
31. Alizadeh, M.; Yang, S.; Sharif, M.; Katti, S.; McKeown, N.; Prabhakar, B.; Shenker, S. Pfabric: Minimal near-optimal datacenter transport. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 435–446. [[CrossRef](#)]
32. Han, Y.; Yoo, J.-H.; Hong, J.W.-K. Poisson shot-noise process based flow-level traffic matrix generation for data center networks. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 450–457.
33. Galluccio, L.; Milardo, S.; Morabito, G.; Palazzo, S. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for wireless sensor networks. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 513–521.
34. OMNeT++. Available online: <https://www.omnetpp.org> (accessed on 25 June 2018).
35. Shreedhar, M.; Varghese, G. Efficient fair queueing using deficit round robin. *IEEE/ACM Trans. Netw.* **1996**, *4*, 375–385. [[CrossRef](#)]
36. Qiu, T.; Liu, X.; Li, K.; Hu, Q.; Sangaiah, A.K.; Chen, N. Community-aware data propagation with small world feature for internet of vehicles. *IEEE Commun. Mag.* **2018**, *56*, 86–91. [[CrossRef](#)]
37. Qiu, T.; Chen, N.; Li, K.; Atiquzzaman, M.; Zhao, W. How can heterogeneous internet of things build our future: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2011–2027. [[CrossRef](#)]



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).