Accurate Rate-Aware Flow-level Traffic Splitting

Ning Wu, Ao Tang
School of Electrical and Computer Engineering
Cornell University
Ithaca, USA
nw276@cornell.edu, at422@cornell.edu

***Abstract*—This paper aims to accurately realize given traffic split ratios in switches with small performance degradation. For given traffic split ratios calculated mathematically by TE algorithms in the control plane, the load distribution mechanisms in the data plane implement such splits without breaking flows. Treating all flows equally, the state-of-the-art approaches deployed in switches do not provide enough accuracy especially when facing non-uniform flow size distribution. We instead propose a dynamic load distribution scheme based on the collected load sharing statistics. It finds the most accurate traffic splits with minimum route changes. We implement our solution in Open vSwitch (OVS). Trace-driven and end-to-end experiments demonstrate that 1) our approach effectively adjusts load distribution in real time to mitigate the inaccuracy of splits caused by the variation of flow size distribution, 2) it outperforms the existing approaches with respect to both higher accuracy and lower level of route changes, and 3) it requires path changes for less flows when routing strategies are reconfigured, hence leads to better flow experience such as higher goodput.**

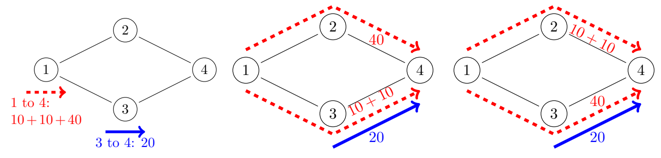
I. Introduction

Traffic Engineering (TE) targets at controlling routing and bandwidth to achieve high reliability, utilization and performance in computer networks [13], [5], [10], [17]. Based on demand information, network state and other operational constraints, routes are computed periodically in the control plane by certain TE algorithms. The solutions are then realized in the data plane by properly updating forwarding tables in switches. For example, multipath TE in general can provide better load balancing across the network. Once TE determines the optimal paths for each commodity flow [1], the corresponding split ratios associated with each outgoing interface at each switch are then set to split the traffic over multiple paths as the solution requires.

There are however two important issues involved in the above described picture. First, how well can a switch realize given traffic split ratios? TE algorithms usually take commodity traffic demand or aggregate link loads as input and then solve certain mathematical optimization problems assuming traffic can be split arbitrarily. On the other hand, in practice, individual flows are not split over multiple paths to avoid potential out-of-order arrivals which can cause significant performance degradation such as TCP goodput drop. This flow-level granularity constraint can well force the actual split ratios deviate from the ones that are demanded by TE. This is particularly true when flows have vastly different rates while current solutions treat all flows equally when it comes to decide which flows should be moved to another path.

[1]The exact meaning of commodity depends on the granularity of the routing schemes. Usually each commodity corresponds to traffic with the same source-destination pair, or just with the same destination subnet.

Such inaccurate load distribution deviating from target TE solutions can result in negative effects such as network performance degradation due to imbalanced resource utilization or even link capacity violation. We here illustrate such consequences with a simple example (Fig. 1). The demand matrix consists of two commodities of traffic: 60 units from node 1 to node 4 and 20 units from node 3 to node 4 (Fig. 1a). Considering the objective of TE as to minimize the maximum link load, one optimal routing scheme (Fig. 1b) is to forward $2/3$ of the traffic from node 1 to node 4 through the upper path and $1/3$ through the lower path. Suppose the traffic are divided into three individual flows based on 5-tuple hash. Being unaware of the flow sizes, path selection module may assign 2 flows with size 10 to the upper path (Fig. 1c). The resulting link load imbalance deviates from the objective of TE and can violate the link capacity constraint if the capacity is no more than 60 units.



(a) Demand (b) Optimal solutions (c) Actual results

Fig. 1. Example of inaccurate splitting.

Second, how much performance cost does realizing traffic splits incur? Here we are not discussing the cost that are caused by the control plane, such as transient loops, but focusing on the cost that is associated with moving a flow from one path to another in the data plane, which can lead to out-of-order packet arrivals when those two paths have different latency. Fig. 2 demonstrates an example of such packets reordering. Initially at time $t = t_0$, packets starting with sequence number 1 were forwarded through the lower path from node 1 to node 4 (Fig. 2a). After $10\,\text{ms}$, its routing path was changed from the lower path to the upper one. Packets with sequence number 1 to 1000 were already sent out during the past $10\,\text{ms}$ and were traveling on the lower path (Fig. 2b). It took around $20\,\text{ms}$ for the first packet on the upper path with sequence number 1001 to reach the destination. At time $t = t_0 + 30\,\text{ms}$, the packets with smaller sequence number were still on the way between node 2 to node 4 (Fig.2c), so out-of-order packets were encountered at the receiver.

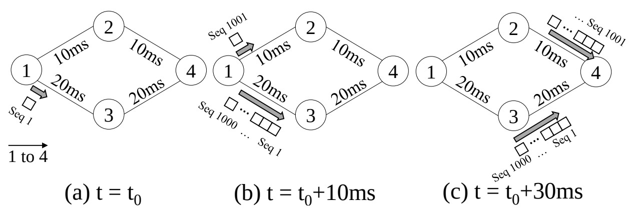


Fig. 2. Example of out-of-order packets.

As traffic demand rapidly grows in both volume and variation, to meet with the Quality of Service (QoS) requirements at high data rates, TE techniques need to become finer, both in terms of adjustment precision as well as update frequency. These in term directly require better solutions in the data plane to deal with the above mentioned two issues, i.e., load balancing mechanisms that can produce traffic splits that are close to what TE algorithms demand with small update cost. Finally, it is worth noting that these two requirements are of competing nature since updating to new traffic splits more precisely usually needs to shuffle more flows. This paper incorporates such tradeoff in a weighted sum optimization.

The rest of the paper is organized as follows. After providing necessary background on how load balancing over multipath is done in the data plane (Section II), we state design constraints, formulate our problem and provide a solution (Section III). We then proceed to implement the solution in Open vSwitch (OVS) (Section IV) and evaluate its performance through trace-driven and end-to-end tests (Section V). A brief summary of related existing work is provided by the end of this paper (Section VI).

II. Background

The general process of forwarding traffic over multiple paths is shown in Fig. 3, which mainly comprises of two key components: traffic division and path selection. The arrival packets are first classified into traffic units in the traffic division module. The outgoing interface for each traffic unit are then determined independently by the path selection module, while all packets from the same traffic unit follow the path identically.

The aggregated traffic are divided into units at a certain level of granularity [12]. The smallest scale of division is a single packet. Since the path for each packet is determined independently, packet-level division achieves the finest-grained splitting but will cause the packet-reordering problems that severely degrades the throughput of transport layer protocols such as TCP. To preserve packet ordering, flow-level traffic division is widely used. In flow-level division, packets that share the same value of some fields in IP packet header are grouped together as a flow with a unique identifier. The granularity of flow unit depends on the fields to be matched in

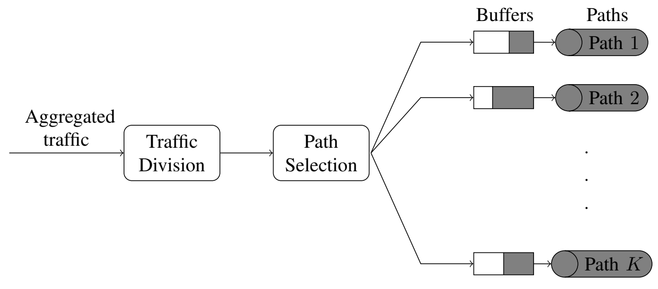


Fig. 3. Multipath forwarding system

the packet header. Packets from a particular flow can be further grouped into subflows, referred to as flowlets, by considering the inter-arrival time of the packets [6].

The path selection module selects the path for each traffic unit independently. The path is determined based on the target split ratios with or without additional information of the traffic load. Being unaware of the traffic load information, the path selector either selects the path in a round robin manner for individual packets which is not desirable due to packet reordering issues, or performs the hash-based path selection for flows/subflows assuming a uniform flow size distribution. This type of path selection schemes in general has low computational complexity and overhead. It is, however, not able to control the actual split ratios when the flow size distribution is skewed. The other type of schemes collects the traffic load information and dynamically adjusts the existing paths assignment in real time to make the actual split ratios close to the target ones.

In many today's commodity routers, equal-cost-multipath (ECMP) is implemented which splits traffic evenly over each path. Weight-equal-cost-multipath (WCMP) is offered by some type of routers with a coarse granularity of split ratios being supported. Those models being implemented nowadays do not collect the information on traffic or network condition in the path selection process, and therefore cannot provide accurate traffic splitting over multiple paths.

III. Formulation and Solution

In this section, we discuss our design choices in detail. Those then become the constraints of our mathematical formulation. An algorithmic solution to the formulation is provided and will be mapped back to the system in the next section. We focus on the flow-level path reassignment problem for one commodity and assume that there are multiple paths available with corresponding target split ratios provided by TE.

A. Key Features

Fig 4 illustrates our design of traffic splitting mechanism. In this subsection, we first present some important design decisions and key features.

Table-based hashing: Our flow-level traffic splitting approach bases on table-based hashing. A flow is first identified by five-tuple in the hash function H and then assigned to a bin among M bins according to its hash value. Each bin is mapped to one of the L paths. We choose table-based hashing for three main reasons: 1) It provides the flexibility of remapping between bins and paths; 2) Path reassignment is scalable because the number of bins is limited and does not change as the number of flows increases; 3) Table-based hashing makes it possible to separate traffic with different routing rules required by different classes of service of priorities. For example, if the high priority traffic are required to be forwarded strictly through the single shortest path, then they need to be excluded from the multipath routing process. It can be realized by assigning this particular type of traffic to a dedicated set of bins which do not allow bin-path remapping.

Rate-aware dynamic path reassignment: The path selection process determines the remapping between bins and paths. Dynamic remapping is proposed to minimize the real-time error between the actual load distribution and the target split ratio for each path. We effectively reassign some of the bins to different paths with the knowledge of the current load sharing statistics.

Direct control of reassignment level: In order to quantitatively control the degree of packets reordering caused by path reassignment, we consider a limit for the amount of traffic that are shifted to a different path.

Valuable reassignment only: In addition to the limited level of reassignment, we also consider a weighting parameter to evaluate the tradeoff between splitting error and the reassignment level. A certain level of remapping is adopted only if it is valuable. We target at finding the best strategy to minimize the error of splitting with only necessary and worthwhile path reassignments.

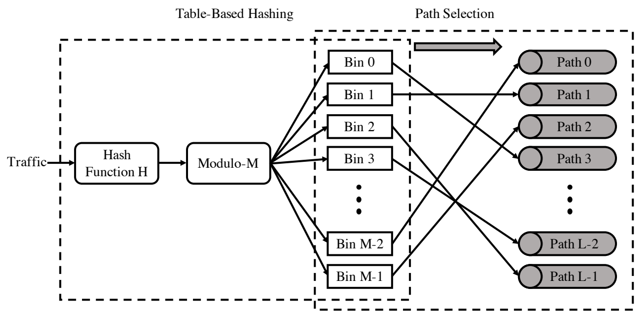


Fig. 4. Design of traffic splitting.

B. Problem Formulation

We now formulate the problem of bin-path remapping. Suppose there is a set $\mathcal{M}$ of M bins being assigned to a set $\mathcal{L}$ of L different paths. Let r_i be the traffic arrival rate of bin i. Let α_l be the target split ratio of path l that is provided by TE solution. x_i^l denotes the mapping decision between bin i and path l, and we have $x_i^l = 1$ if bin i is assigned to path l and 0 otherwise. We further define the overflow fraction ρ_l to be the difference between actual load fraction and the target split ratio for path l, given by $\rho_l = \frac{\sum_{i\in\mathcal{M}} r_i x_i^l}{\sum_{i\in\mathcal{M}} r_i} - \alpha_l$. Therefore the maximum overflow fraction among all available paths is $\rho = \max_{l\in\mathcal{L}}\{\rho_l\}$. Assuming the current assignment strategy is determined by a set of $\{x_{i,0}^l\}$, we aim at finding the optimal $\{x_i^l\}$ that minimizes the weighted sum of the maximum overload fraction ρ and the degree of reassignment fraction. The weighting parameter is denoted as λ. To quantify the degree of reassignment, we define Δ to be the number of bins being remapped. We add an additional constraint that the number of bins allowed to be reassigned is limited by K. Putting all together, the problem formulation is given by

$$\min \quad \rho + \lambda\Delta \tag{1}$$
$$\text{s.t.} \quad \sum_{i\in\mathcal{M}} r_i x_i^l \le (\rho+\alpha_l)\sum_{i\in\mathcal{M}} r_i \quad \forall l\in\mathcal{L} \tag{1a}$$
$$\sum_{l\in\mathcal{L}} x_i^l = 1 \quad \forall i\in\mathcal{M} \tag{1b}$$
$$x_i^l \in \{0,1\} \quad \forall i\in\mathcal{M}, l\in\mathcal{L} \tag{1c}$$
$$M - \sum_{l\in\mathcal{L}}\sum_{i\in\mathcal{M}} x_i^l x_{i,0}^l = \Delta \tag{1d}$$
$$\Delta \le K \tag{1e}$$

In the objective function, the parameter λ determines the tradeoff between maximum overflow and the number of bins being reassigned. Condition 1a defines the maximum overflow fraction ρ such that the overflow fraction for each path should not exceed. Condition 1b and 1c require that one bin can only be assigned to one path. Condition 1d describes the number of bins that are selected to be remapped. Condition 1e restricts the number of bins to be remapped. One underlying rule based on the objective function is if the initial maximum overflow ρ_0 satisfies $\rho_0 \le \lambda$, then the optimal value is $opt^* = \rho_0$ with $\Delta^* = 0$, meaning that we prefer to not reassign any bins because any path change costs more price than benefits. Another intuitive rule following that is any optimal solution of Δ^* satisfies $\Delta^* \le \rho_0/\lambda$.

Our problem formulation differs from the canonical bin-packing problem in two main aspects. Firstly, we consider the price of the bin reassignment, and the solutions are constrained by the number of bins being reallocated. Furthermore, overflow is mostly possible in our reallocation strategy while it is not allowed in the original bin-packing problem.

C. Algorithm

In this subsection, we propose an efficient algorithm that computes the approximate reassignment solutions to the optimization problem. The idea behind the algorithm is to first estimate a lower bound of the maximum overflow fraction, to position the goal of accuracy at an achievable and reasonable level with the limit on the reassignment degree. Based on the lower bound, we then select the set of bins from a candidate pool and reassign them to new paths.

1) Lower bound of the maximum overflow:
We obtain a lower bound of the maximum overflow fraction by finding the optimal solution to the following Linear Programming (LP) relaxation:

$$\begin{aligned}
\min\ & \rho && (2)\\
\text{s.t.}\ & \sum_{i\in\mathcal{M}} r_i x_i^l \le (\rho+\alpha_l)\sum_{i\in\mathcal{M}} r_i && \forall l\in\mathcal{L}\\
& \sum_{l\in\mathcal{L}} x_i^l = 1 && \forall i\in\mathcal{M}\\
& x_i^l \in [0,1] && \forall i\in\mathcal{M}, l\in\mathcal{L}\\
& \sum_{i\in\mathcal{M}}(1-\sum_{l\in\mathcal{L}} x_i^l x_{i,0}^l) r_i \le \delta \sum_{i\in\mathcal{M}} r_i
\end{aligned}$$

$\delta = K/M$ is the limit of the reassigned load fraction, given the limited number of reassigned bins K in problem (1). Let ρ_l^0 be the initial overflow fraction for path l, i.e., $\rho_l^0 = \frac{\sum_{i\in\mathcal{M}} r_i x_{i,0}^l}{\sum_{i\in\mathcal{M}} r_i} - \alpha_l$. Assuming the paths are sorted in the descending order of ρ_l^0, we then have the following

Theorem 1. *The optimal value ρ^* of the LP relaxation (2) is*

$$\rho^* = \max\{\frac{\sum_{l=1}^{\hat{l}} \rho_l^0 - \delta}{\hat{l}}, 0\}, \qquad (3)$$

where $\hat{l}$ is defined as

$$\hat{l} = \min\{i \in \mathcal{L} : \frac{\sum_{l=1}^{i} \rho_l^0 - \delta}{i} \ge \rho_{l+1}^0\}. \qquad (4)$$

Proof. First note that the sum of overflow fraction for all paths is zero because $\sum_{l\in\mathcal{L}} \rho_l = \frac{\sum_{l\in\mathcal{L}}\sum_{i\in\mathcal{M}} r_i x_i^l}{\sum_{i\in\mathcal{M}} r_i} - \sum_{l\in\mathcal{L}} \alpha_l = 0$. Hence by definition ρ is nonnegative. Let $\mathcal{L}^+$ be the set of paths being overloaded initially, i.e. $\rho_l^0 > 0, \forall l \in \mathcal{L}^+$, and assume $\mathcal{L}^+$ is a sorted set in the descending order of ρ_l^0. Similarly, let $\mathcal{L}^-$ be the sorted set of paths being underloaded initially.

a) If $\sum_{l\in\mathcal{L}^+} \rho_l^0 \le \delta$, then $\frac{\sum_{l=1}^{\hat{l}} \rho_l^0 - \delta}{\hat{l}} \le 0$ for $\forall j \in \mathcal{L}$. Hence by Eq. 3, $\rho^* = 0$. This can be easily achieved with total reassignment fraction no larger than δ by moving exactly ρ_l^0 load fraction on each path $l \in \mathcal{L}^+$ to paths in $\mathcal{L}^-$.
b) If $\sum_{l\in\mathcal{L}^+} \rho_l^0 > \delta$, then by Eq. 4, $\hat{l}$ is no less than the last-indexed path in $\mathcal{L}^+$. So by Eq. 3 $\rho^* = \frac{\sum_{l=1}^{\hat{l}} \rho_l^0 - \delta}{\hat{l}} > 0$. It can be achieved by shifting load from each path $l = 1, \cdots, \hat{l}$ to paths in $\mathcal{L}^-$ with the amount of $\rho_l^0 - \rho^*$. We prove its optimality by contradiction. Suppose there exists a feasible solution resulting in $\rho' < \rho^*$, then the reassigned load fraction is

$$\sum_{l=1}^{\hat{l}} \rho_l^0 - \hat{l}\rho' > \sum_{l=1}^{\hat{l}} \rho_l^0 - (\sum_{l=1}^{\hat{l}} \rho_l^0 - \delta) = \delta.$$

This contradicts the constraint of reassigned load limit δ. Therefore,

□

We here present a procedure to construct a set of reassigned bins and show that it leads to an optimal solution to the LP relaxation (2).

Procedure 1:

(i) Starting from the first path l in $\mathcal{L}^+$, we define a critical bin s_l for path l such that $s_l = \min\{i \in B_l : \sum_{j=1}^{i} r_j > (\rho^* + \alpha_l)\sum_{i\in\mathcal{M}} r_i\}$, where B_l is the set of bins initially assigned to path l.
(ii) For $\forall i \in B_l$, we set the variable x_i^l to be

$$x_i^l = \begin{cases} 1 & \text{if } i < s_l,\\ \frac{(\rho^*+\alpha_l)\sum_{i\in\mathcal{M}} r_i - \sum_{j=1}^{s_l-1} r_j}{r_{s_l}} & \text{if } i = s_l,\\ 0 & \text{otherwise.}\end{cases}$$

(iii) Repeat Step (i) and (ii) for each $l \in \mathcal{L}^+$ until $\hat{l}$. Define a set D for partially or completely disconnected bins such that $D = \{i \in B_l : x_i^l < 1, \quad l = 1, \cdots, \hat{l}\}$.

Theorem 2. *The set of reassigned bins obtained by Procedure 1 can result in the optimal value ρ^* of the LP relaxation (2).*

Proof. The current load for path $l = 1, \cdots, \hat{l}$ after Procedure 1 is exactly $(\rho^* + \alpha_l)\sum_{i\in\mathcal{M}} r_i$, and thus we have that the current overflow fraction $\rho_l = \rho^*$ for $l = 1, \cdots, \hat{l}$. According to Eq. 4, $\rho_l^0 \le \rho^*$ for $l = \hat{l}+1, \cdots, L$. Given that $\sum_{l\in\mathcal{L}} \rho_l^0 = 0$ and the reassignment fraction is no larger than δ, we have

$$\sum_{l\in\mathcal{L}^-} \rho_l \le \sum_{l\in\mathcal{L}^-} \rho_l^0 + \delta \le -\sum_{l=1}^{\hat{l}} \rho_l^0 + \delta = -\sum_{l=1}^{\hat{l}} \rho_l = -\hat{l}\cdot\rho^* \le 0$$

Hence there always exists a remapping solution between the bins in D and paths in $\mathcal{L}^-$ such that $\rho_l \le 0$ for all $l \in \mathcal{L}^-$. So the maximum overload fraction among all path $l \in \mathcal{L}$ is ρ^*. Based on Theorem 1, ρ^* is the optimal value of the LP relaxation (2). Therefore, the set of reassigned bins following Procedure 1 can lead to the optimal value of the LP relaxation (2). □

2) Bins selection and reassignment:
The optimal value ρ^* provides a lower bound for the optimization problem (1), guiding us to target at the reasonable level of accuracy without unnecessary reassignment. Let D^* be the set of bins that we select to be remapped. We follow the procedure described above to find the set of disconnected bins D and put the items from D into D^* excluding the sets of critical bin s_l for each path $l = 1, \cdots, \hat{l}$. Then we compute the current overflow fraction ρ_l for path $l = 1, \cdots, \hat{l}$ and sort the set of path in descending order in terms of ρ_l. We select one bin from each of these paths to be further put into D^* until the number of reassigned bins exceeds K. Once D^* is complete, we use Best Fit algorithm to reconnect the bin set D^*. Algorithm 1 illustrates the pseudocode of our algorithm.

D. Complexity

The time complexity of sorting set $\mathcal{L}$ with L paths is $O(LlogL)$. For a set of $\mathcal{M}$ with M bins and K maximum

```
Input: $\{x_{i,0}^l\}$, $\{\alpha_l\}$, $\{r_i\}$, $\rho_0$, $\lambda$, K, $\forall i \in \mathcal{M}, \forall l \in \mathcal{L}$
Output: $\{x_i^l\}$
begin
    $opt = \rho_0$, $D^* \leftarrow \emptyset$
    for $j = 1 \mathbf{to} K$ do
        $\delta \leftarrow j/M$;
        Find $\rho^*$ and $\hat{l}$ using Eq. 3 and Eq. 4;
        Get $D$ and $\{s_l\}$ by following procedure (i) - (iii);
        $D^* \leftarrow D \setminus \{s_l\}$;
        Compute $\{\rho_l\}$ for $l = 1, \cdots, \hat{l}$;
        foreach $l \in \{\rho_l\}$ sorted in descending order do
            Find bin $b_t$ in path $l$ such that
            $|\rho_l - \rho^* - r_t / \sum_{i \in \mathcal{M}} r_i|$ is minimum;
            if $Sizeof(D^*) + 1 \leq K$ then
                $D^* \leftarrow D^* \cup b_t$;
            else
                Break;
            end
        end
        Reassign bins in $D^*$ among all paths in $\mathcal{L}$ using
        Best Fit algorithm;
        Compute $\{x_i^l\}_j$ and $\rho_j$;
        if $\rho_j + \lambda \cdot Sizeof(D^*) < opt$ then
            $opt \leftarrow \rho_j + \lambda \cdot Sizeof(D^*)$;
            $\{x_i^l\} \leftarrow \{x_i^l\}_j$;
        end
    end
end
```

Algorithm 1: Pseudocode.

number of reassigned bins, the runtime complexity of bins selection and reassignment is $O(M + KlogK)$. So the worst-case time complexity of our proposed algorithm is $O\big(LlogL + K(M + KlogK)\big)$. Note that the computation complexity is not increased with the amount of flows, and thus the splitting scheme is scalable. L is determined by the number of available paths in the network. The value of M and K affect the granularity of the traffic splitting. With the reasonable values of M and K to achieve desired accuracy in practice, both the computation complexity and the implementation complexity of the algorithm is marginal.

IV. Implementation

Open vSwitch (OVS) is used here as the platform for implementation. We will first explain how traffic splitting is done in OVS including why it cannot provide accurate splits even when all flows are of the same size (Section IV-A). Implementation details are provided in Section IV-B.

A. Splitting Approach in OVS

The weighted traffic splitting is realized in group table in OVS. Multiple buckets with corresponding weights can be added to one entry of the group table. Buckets are associated with specific actions such as forwarding the packets to a particular output port, so that the traffic splitting among multiple outgoing links/paths is realized by the bucket selection process. The bucket selection in OVS group selects the bucket on flow level identified by the 5-tuple.

The algorithm of flow-level weighted splitting implemented in OVS is the following. Suppose one group table entry contains L buckets. Let b_i and w_i denote the bucket id for the ith bucket and its corresponding weight, $i = 1, 2, \cdots, L$. When processing one packet p_k belonging to this group, it first obtains its hash value $a_k = H(p_k)$ by applying hash function H with 5-tuple as the input; then gets a hash value associated with each bucket $\beta_i^k = M(b_i, a_k)$ via hash function M; finally selects the bucket with the highest hash value of β_i^k.

We would like to point out that the OVS's approach cannot produce accurate ratios consistent with the weights, regardless of the quality of hashing function, the amount and heterogeneous of flows. To see that, consider the case of two buckets. Assuming there are enough amount of flows, the hash function $M(\cdot)$ for the two buckets can be viewed as choosing two independent integer random variables from the interval $[0, 2^{32} - s]$. We now compute the probability that the value of $\hat{\beta_1^k} \times w_1$ is greater than the value of $\hat{\beta_2^k} \times w_2$. Let X and Y be the two independent random variables from the interval $[0, a]$ and $[0, b]$ respectively with uniform probability density. The problem to be solved becomes computing $Prob(X - Y > 0)$, the probability that X is larger than Y. Let $f_X(x)$, $f_Y(y)$, $f_Z(z)$ denote the density functions for X, Y, and $Z = X - Y$ respectively. We have

$$f_X(x) = \begin{cases} \frac{1}{a} & 0 \leq x \leq a, \\ 0 & \text{otherwise.} \end{cases}$$

$$f_Y(y) = \begin{cases} \frac{1}{b} & 0 \leq y \leq b, \\ 0 & \text{otherwise.} \end{cases}$$

$$f_Z(z) = \int_{-\infty}^{+\infty} f_X(z+y) f_Y(y) dy = \frac{1}{b} \int_0^b f_X(z+y) dy$$

Because $f_X(z + y)$ is 0 unless $0 \leq z + y \leq a$, i.e. $-z \leq y \leq a - z$, assuming $a > b$, we have

$$f_Z(z) = \begin{cases} \frac{1}{ab} \int_{-z}^{b} dy = \frac{b+z}{ab} & -b \leq z \leq 0, \\ \frac{1}{ab} \int_0^b dy = \frac{1}{a} & 0 \leq z \leq a - b, \\ 0 & \text{otherwise.} \end{cases}$$

We can easily compute $Prob(X - Y > 0) = Prob(Z > 0) = \frac{2a-b}{2a}$, which is not equal to $\frac{a}{a+b}$ unless $a = b$.

One optional revision with minimum modifications of the code is to select b_I such that $I = i : \max_j (\frac{\sum_{l=1}^{j} w_l}{\sum_{l=1}^{N} w_l} < \frac{\alpha_k}{2^{32}-1})$, which will be considered as a compared scheme in Section V.

B. Implementation

The implementation of the traffic splitting mechanism is broken down into three components: initial path assignment, load statistics update and bin reallocation. Our implementation is based on the original mechanism of group table's weighted buckets in OVS with minimum modification in user space. Under the structure of a group, we insert a list of 32 bins. Each bin is associated with a bin id, a counter and a bucket.

Initial path assignment: When the first packet of a flow arrives, it is assigned to a specific bin. If the bin is already mapped to a bucket, then packets from this flow are forwarded through the path specified in the bucket. Otherwise, a bucket having the largest underload will be assigned to the bin.

Load statistics update: Our dynamic traffic splitting relies on the updated statistics from each bucket and bin. In OVS the datapath in user space polls the accumulated number of bytes information for each flow from the kernel space once every 0.5s. Our load statistics update requires the minimum implementation complexity by only additionally updating the counter of each bin according to the flow's statistics information. Then the rate (bytes per second) for each bin and each bucket are calculated.

Bin reallocation: The function of bin reallocation can be triggered by two events. A timer fires the bin reallocation process once every T seconds. It aims at improving the accuracy of splitting for existing flows. In addition, it can be triggered immediately by any reconfiguration of target split ratios caused by TE routing re-computation. Once the bin reallocation process is triggered, it first collects the load information from bins and buckets, as well as the current bin assignment. Then either the ILP problem (1) is structured and solved by the GLK solver, or the implemented algorithm described in Section III-C will be called. The output as the new assignment strategy will be configured.

V. Performance Evaluation

A. Evaluation Environment

Testbed setup: We build a network topology in Mininet consisting of two source subnets S_1 and S_2, one destination subnet D_1 and four paths between them as shown in Fig. 5. Each subnet represents multiple hosts. The propagation delay (millisecond) is labeled on the corresponding link. The bandwidth of each link is $100\,\mathrm{Mbps}$. OVS with the implementation of our traffic splitting mechanism is used as the virtual switch.

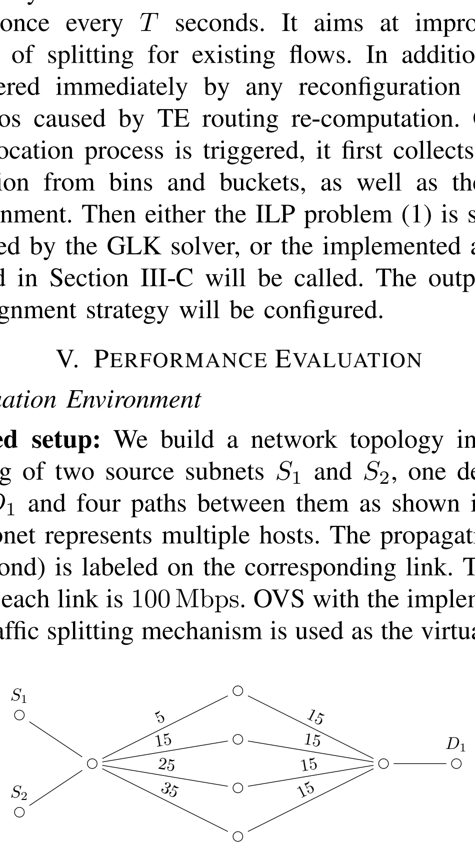


Fig. 5. Network topology.

Traffic trace: To study the flow-level Internet traffic characteristics, we analyze the statistics of traces collected from CAIDA [1] and WIDE [2]. We select three destination subnets and extract the corresponding packets for 60 seconds. Before the performance evaluation, we would like to verify, based on the traces, that the inaccuracy of splitting indeed exists and there is enough room to improve the accuracy. It was noted in [3], [14] that the skewness of flow size distribution is the main cause of splitting inaccuracy. The three traffic traces have different mean flow size and mean throughput as illustrated in Table I, but their flow size distribution and flow arrival rate distribution follow a similar pattern. Fig. 6a shows that for each of the trace, over 70% of the flows have size less than the mean flow size, while the maximum flow size could be as large as 2000 times of the mean (for trace 1 and 2). The small portion of super large flows makes it hard for static traffic splitting to perform accurately. We also plot the cumulative distribution function of flow's arrival rate in Fig. 6b. The x-axis shows the ratio of the flow arrival rate to the mean throughput. Note that the mean throughput of a trace is the aggregated arrival rates for all flows over 60 seconds, and thus it is possible for a particular flow's arrival rate to exceed the mean throughput. We observe that for each trace there exists a small number of flows having arrival rates higher than 5% of the mean throughput.

We replay the traffic trace using the tool tcpreplay which reproduces the packet-level synthetic traffic of the collected trace, both in terms of packet arrival time and packet size. We use tcprewrite to overwrite the IP Address and Ethernet Address of the packet's header with the appropriate addresses in our emulation network. The port numbers on the transport layer are kept unchanged in order to maintain the entire flow characteristics of the original trace. The traffic replay are injected into the network from the source host.

Testing scenarios: Our experiments are conducted in two scenarios. The path reassignment is necessary and valuable in either of the two cases: 1) when facing non-uniform flow size distribution, or 2) when the target split ratios are changed. In the first scenario, we assume the target split ratios are not updated so that our reassignment process dynamically improves the accuracy of splitting caused purely by the existing flows in the network. In the second scenario, we take into account the update of target split ratios which is reconfigured by dynamic TE and evaluate how our scheme reacts in real time.

Compared splitting schemes: We consider 5 splitting schemes in our evaluation.
RA-alg: the algorithmic approach to our rate-aware traffic splitting proposed in III-C.
RA-opt: the approach that finds the optimal solution by solving the optimization problem (1).
OVS: the original weighted-splitting mechanism in group table implemented in OVS.
OVS-revised: the revised algorithm for OVS we described in Section IV-A.
MBD-/ADBR: a representative scheme of dynamic traffic splitting based on the load statistics [9]. It first disconnects multiple bins in the order of decreasing size from each overloaded path until all paths are underloaded, and then reconnects each of the disconnected bins to the path with the largest underloaded.

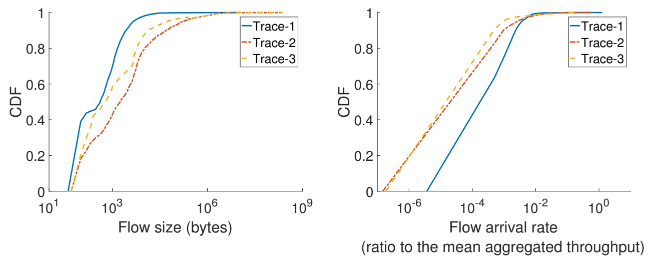
Performance metrics:

The maximum overflow ρ and reassignment fraction Δ are considered as the two primary performance metrics. ρ is obtained by finding the maximum overflow fraction among the four paths. The reassignment fraction Δ is measured by summing up the fraction of traffic being shifted to a different

TABLE I
FLOW-LEVEL PROFILE OF TRAFFIC TRACES

Trace ID	# total flows	Flow size (Kbytes)			Mean throughput (Mbps)	Flow arrival rate (Kbps)			# flows having arrival rate larger than	
		Mean	Min.	Max.		Mean	Min.	Max.	5% of mean throughput	10% of mean throughput
1	6611	4.96	0.044	10655	4.37	5.46	0.016	5483.8	9	5
2	8403	128.04	0.06	233869	143.45	133.68	0.021	95946.4	19	6
3	3993	166.54	0.06	52069	88.67	163.16	0.017	17039.3	55	22

path at each time when the reassignment is triggered. We also consider goodput and packet re-ordering fraction to evaluate the network performance. Goodput measures the application-level throughput that excludes retransmitted packets. Packet reordering fraction is the percentage of packets that do not advance the sequence number when arriving at the destination.



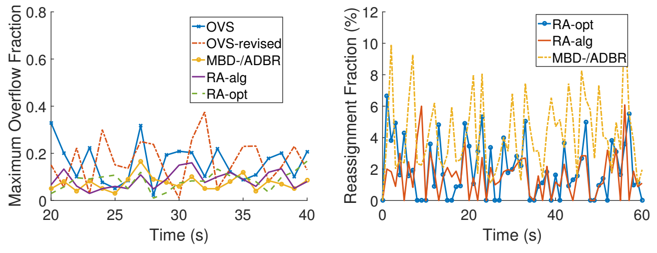
(a) (b)

Fig. 6. CDF of flow size and flow arrival rate for packet traces.

B. Constant Splits

Equal splitting: We first evaluate the performance for equal splitting which targets at splitting the traffic among the four available paths evenly. The desired split ratio for each path is 0.25. We replay the three 60-second traces and send all traffic from the source host subnet S_1 to the destination host subnet D_1. In RA-opt and RA-alg, we set λ to be 0.05 and K to be 6. The update time interval for RA-opt, RA-alg and MBD-/ADBR are all 1 second.

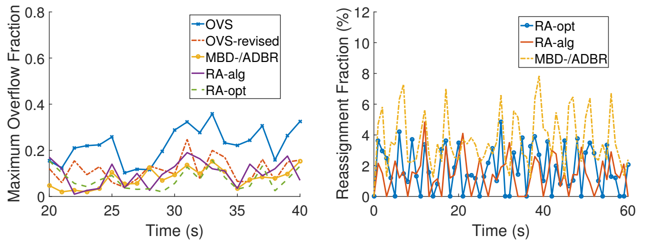
Fig. 7 shows the maximum overflow fraction and reassignment fraction over time for trace 1. The average performance of accuracy among all three traces are shown in Fig. 9. OVS and OVS-revised do not readjust the paths for existing flows. Due to the flow dynamics and skewed flow size distribution, the maximum overflow in these two schemes can be up to 0.4, as shown in Fig. 7a. The other three schemes dynamically readjust the path assignment based on the current load statistics, so the maximum overflow is significantly reduced when they are used. Fig. 7b shows the fraction of load being reassigned to different paths for MBD-/ADBR, RA-opt and RA-alg. MBD-/ADBR reassigns as much as 4.51% of total load on average because it redistributes the traffic load in a greedy manner and involves unnecessary reassignment. RA-opt redistributes the load based on the optimal solution, so it minimizes the maximum overflow with only around 2.11% load being reassigned. Compared to the optimal solution provided by RA-opt, RA-alg is more conservative and shows the least reassignment fraction of 1.95%. It still achieves similar level of accuracy as RA-opt and MBD-/ADBR but moves much fewer flows to different paths than MBD-/ADBR.



(a) Maximum overflow fraction. (b) Reassignment fraction.

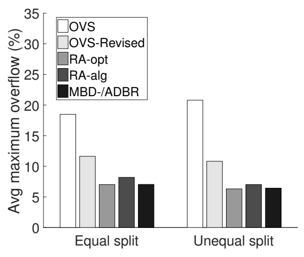
Fig. 7. Equal splitting.

Unequal splitting: We next consider unequal splitting amongst the four paths with split ratio 0.1: 0.2: 0.3: 0.4. As we discussed in Section IV-A, the current splitting algorithm implemented in OVS fails to perform correctly for weighted splits even when the flow sizes are homogeneous. This is verified in Fig. 8a and Fig. 9 which show much larger maximum overflow and split ratio deviation for OVS than OVS-revised in the case of unequal splitting. We observe the similar performance comparison results as the equal splitting case: RA-opt achieves the lowest maximum overflow with around 1.87% reassignment fraction; the splitting accuracy for MBD-/ADBR is close to RA-opt but causes 3.15% reassignment on average; RA-alg has the lowest degree of reassignment which is about 1.75% and a bit larger maximum overflow and deviation of load distribution than RA-alg and MBD-/ADBR.

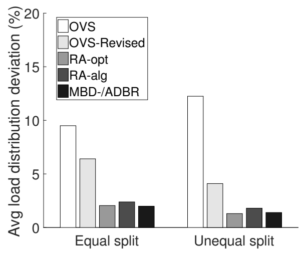


(a) Maximum overflow fraction. (b) Reassignment fraction.

Fig. 8. Unequal splitting.



(a)



(b)

Average reassignment fraction (%)

	RA-opt	RA-alg	MBD-/ADBR
Equal split	2.11	1.95	4.52
Unequal split	1.87	1.75	3.15

(c)

Fig. 9. Performance comparison: a. Average maximum overflow fraction; b. Average deviation of load distribution meausres the sum of the load fraction's deviation from each path's target split ratio; c. Average reassignment fraction.

Impact of parameters: Two parameters in our model play important roles in the tradeoff between accuracy and reassignment degree: the reassignment time interval T and the weighting parameter λ. We evaluate their effects on the performance by tuning one parameter while fixing the other. The experiments are based on unequal splitting scenario with RA-opt and the traffic trace replay.

The parameter λ quantitatively determines how much price the reassignment costs when reducing the level the overflow. $\lambda = 0$ means we do not consider any cost of reassignment. The larger λ, the more cautions to pay when making changes to the original assignment. Table II shows that as λ increases, the average maximum overflow and thus the deviation of average load distribution become larger because it is more expensive to change the assignment. The average reassignment fraction consequently is decreased. The splitting accuracy and reassignment fraction is also affected by the reassignment time interval T, as is demonstrated in Table III. Higher frequency helps to achieve smaller splitting errors at the price of causing more load to be reassigned.

TABLE II
IMPACT OF λ WHEN $T = 1\,\mathrm{s}$

λ	0.01	0.05	0.08	0.1
Average maximum overflow (%)	2.02	6.23	7.31	8.75
Average deviation of load distribution (%)	0.63	1.5	2.76	3.21
Average reassignment fraction (%)	8.56	1.87	1.14	0.76

C. Dynamic Splits

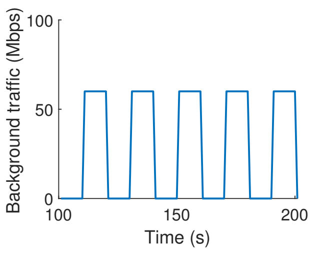
We further evaluate the end-to-end performance when dynamic TE is adopted. We consider the objective of dynamic TE as to balance the loss rate among multiple paths. To simplify the dynamic TE behavior, we use two paths with $40\,\mathrm{ms}$ and $100\,\mathrm{ms}$ round-trip time (RTT) in Fig. 5. Trace-driven experiments capture the characteristics of Internet traffic, but do not suffice to show the end-to-end network performance of traffic splitting schemes. We generate 50 TCP flows via iPerf from different hosts at the host subnet S_1 to the destination subnet D_1. Additional delays ranging from 0 to $20\,\mathrm{ms}$ are randomly added for these flows in order to make the TCP flows heterogenous. Each buffer size is 3000 bytes. We further inject a $60\,\mathrm{Mbps}$ UDP flow as the background traffic at the host subnet S_2 with the dynamic pattern shown in Fig. V-C. Dynamic TE collects the packet loss information for each interface and updates the target split ratios every 2 seconds via the group table API supported by OVS.

TABLE III
IMPACT OF T WHEN $\lambda = 0.05$

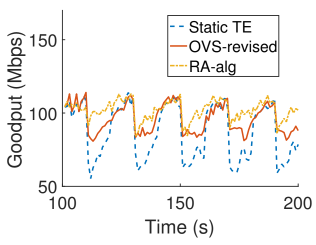
T	0.5s	1s	5s	10s
Average maximum overflow (%)	3.75	6.23	8.52	9.34
Average deviation of load distribution (%)	0.84	1.5	2.97	3.35
Average reassignment fraction (%)	5.46	1.87	0.66	0.29

Fig. 10b and Table IV show the end-to-end performance when dynamic TE works with different traffic splitting schemes. We also compare with static TE which does not react to the real-time loss to verify that dynamic TE indeed brings benefits. The static TE configures constant target split ratios 0.5:0.5 to be realized by OVS-revised in data plane. It is observed in Fig. 10b that when the large background flow is injected, the goodput under static TE scheme shows considerable degradation because it does not move away any TCP flows from the congested path. With dynamic TE, the goodput also drops as soon as the background traffic is added, but is able to climb up gradually since dynamic TE tries to maintain the loss fairness by changing the split ratios. It is further verified in Table IV. The aggregated loss rate sums up the loss rate of the buffer associated with each interface. The static TE has a much larger aggregated loss rate than the dynamic TE. The nonzero packets reordering rate for static TE shown in Table IV is due to the multicore design of OVS.

When new target split ratios are configured by dynamic TE, OVS-revised redistributes the flows accordingly but the error between the actual load distribution and the target ones is not controlled. Therefore, the actual behavior that dynamic TE observes deviates from expected. This triggers the dynamic TE to further adjust the load distribution which makes the target split ratios possibly fluctuate in a large range. That is mainly why the goodput of OVS-revised is lower and its loss rate and packets reordering rate is higher than other splitting approaches, as is shown in Table IV. RA-opt performs the most accurate splitting so the aggregated loss rate is the minimum and it achieves the highest goodput among all schemes. RA-alg has the minimum packets reordering rate, but its goodput is slightly lower than RA-opt because its splitting is not as accurate as RA-opt. MBD-/ADBR invokes more path reassignment than necessary, so high packets reordering rate is the main cause of its goodput degradation.



(a) Background traffic pattern.



(b) Goodput of the TCP flows.

Fig. 10. Goodput when background traffic are added.

TABLE IV
END-TO-END PERFORMANCE

	Static TE	OVS-revised	MBD-/ADBR	RA-alg	RA-opt
Average goodput (Mbps)	86.2	95.6	98.5	102.6	103.4
Aggregated loss (%)	1.837	1.225	1.051	0.998	0.963
Reordering packets (%)	0.374	1.252	1.143	0.805	0.829

VI. RELATED WORK

We classify the existing traffic splitting models into two categories based on whether the current load information are required in their path selection methods.

Load-Unaware:

Packet-By-Packet Round-Robin: The simplest model of info-unware traffic splitting is packet-by-packet load balancing that selects individual packets amongst the alternative paths in round-robin [16] and weighted round-robin fashion [11].

Fast Switching: Cisco proposed fast switching [8] which restricts the size of lookup table by only storing recently seen flows in a cache. When the cache is used up, the oldest entry is deleted in favor of the new one. The performance of splitting varies depending on the size of the cache.

Hash-based: Hash-based approaches identify flow by applying hashing function to packets such that packets from the same flow have an identical hashing value. With the packet's hashing value, direct hashing approaches select the path taking modulo of the number of multiple paths [3]. Its main limitation occurs when a path is added or removed from the original path set, then a certain amount of flows are redistributed.

Masking Operations: [7] and [15] both exploit the bit-masking operations to achieve traffic splitting with finer granularity, relying on the the matching entry feature of OpenFlow.

Load-Aware:

Flowlet Aware Routing Engine (FLARE): [6] proposed FLARE that split a flow into flowlets based on the inter-arrival time of the flow. The timeout of a flowlet forwarding rule is set on the level of path RTT. The amount of traffic being reallocated mainly depends on the parameter of time threshold. In wide area networks, the inter-arrival timeout need to be large enough to maintain low risk of packet reordering, resulting in limited quantity of flowlets and limited room to adjust the load distribution.

Table-based Hashing with Reassignments: [4] performs adaptive load reallocation based on the table-based hashing. The redistribution decision is made by considering both the traffic load information and the inactive time for each bin. Only one bin being remapped at each time interval restricts the improvement of splitting accuracy.

Single/Multiple Bin Disconnection and Reconnection: [9] considers the bin disconnection and reconnection process separately. A set of greedy algorithms are proposed and evaluated that dynamically adjust the load distribution by bin disconnection and reconnection for single or multiple bins either in progressive or conservative manner.

REFERENCES

[1] The CAIDA UCSD Anonymized Internet Traces - February 2012. http://www.caida.org/data/passive/passive_2012_dataset.xml.
[2] Wide backbone traffic traces. http://mawi.wide.ad.jp/mawi/samplepoint-F/2017/201701101400.html.
[3] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. IEEE, 2000.
[4] T. W. Chim, K. L. Yeung, and K.-S. Lui. Traffic distribution over equal-cost-multi-paths. *Computer Networks*, 2005.
[5] S. Kandula et al. Walking the tightrope: Responsive yet stable traffic engineering. In *SIGCOMM*, 2005.
[6] S. Kandula, D. Katabi, et al. Dynamic load balancing without packet reordering. *ACM SIGCOMM Computer Communication Review*, 2007.
[7] N. Kang, M. Ghobadi, et al. Efficient traffic splitting on commodity switches. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 2015.
[8] K.-C. Leung and V. O. Li. Generalized load sharing for packet-switching networks. i. theory and packet-based algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 2006.
[9] R. Martin, M. Menth, and M. Hemmkeppler. Accuracy and dynamics of hash-based load balancing algorithms for multipath internet routing. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*. IEEE, 2006.
[10] N. Michael and A. Tang. Halo: Hop-by-hop adaptive link-state optimal routing. *IEEE/ACM Transactions on Networking (TON)*, 2015.
[11] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM transactions on networking*, 1993.
[12] S. Prabhavat, H. Nishiyama, et al. On load distribution over multipath networks. *IEEE Communications Surveys & Tutorials*, 2012.
[13] A. Shaikh et al. Load-sensitive routing of long-lived IP flows. In *SIGCOMM*, 1999.
[14] W. Shi, M. H. MacGregor, and P. Gburzynski. Load balancing for parallel forwarding. *IEEE/ACM Transactions on Networking (TON)*, 2005.
[15] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou. Flexible traffic splitting in openflow networks. *IEEE Transactions on Network and Service Management*, 2016.
[16] C. Villamizar. Ospf optimized multipath (ospf-omp). *Work in Progress*, 1999.
[17] H. Wang et al. COPE: traffic engineering in dynamic networks. In *SIGCOMM*, 2006.