Bartolomeu: An SDN Rebalancing System Across Multiple Interdomain Paths

Pedro R. Torres Jr[a,*], Alberto García-Martínez[b], Marcelo Bagnulo[b] and Eduardo Parente Ribeiro[a]

[a] *Universidade Federal do Paraná, Centro Politécnico, 81531-890, Curitiba, Brazil*
[b] *Universidad Carlos III de Madrid, Av. de la Universidad 30, 28911, Madrid, Spain*

ARTICLE INFO

Keywords:
Multipath
Interdomain
Load Balancing
BGP
SDN
Flows

ABSTRACT

Bartolomeu is a solution to enable stub networks to perform adaptive egress traffic load balancing across multiple interdomain routes by spreading the traffic across available paths according to a passive measurement of their performance. It defines a BGP-SDN architecture that increases the number of BGP routes that can be used by stub networks. Bartolomeu measures the available capacity of each path to any destination prefix, and allocates to each path a number of large flows that is proportional to its capacity. This strategy reduces the mean sojourn time, i.e., mean time to flow completion, compared to state-of-the-art traffic balancing techniques as ECMP. We develop a mathematical model to compute this time and compare with ECMP and single path (fast path) selection. An analysis of the traffic traces of two content providers was performed to ensure that our solution is deployable. An experiment with traffic exchange over the Internet is used to show that Bartolomeu can provide gains with real interfering traffic. A discrete-event simulator fed with the traces captured is used to asses Bartolomeu's gains with prefixes with different number of flows, and flows with different sizes and arrival time. We observe in this experiment that Bartolomeu can reduce the sojourn time, compared to ECMP, by half when path rates differ in a factor of 3, or to a sixth when path rates differ in a factor of 10. We compute the maximum number of per-flow entries and the maximum entry change request rate to show that the resources required fit in with the specifications of the current generation of SDN switches.

1. Introduction

Today's Internet is built upon an increasingly dense topology, featuring a large number of networks and interconnections. BGP, the *Border Gateway Protocol*, is used to exchange routing information between different networks, called *Autonomous Systems*, and thus generates routes to every destination.

Despite the incredible success of the interdomain routing system, current networks are not able to adapt traffic forwarding to the current path conditions. To illustrate this statement, we consider the case of a content provider network serving large files, such as Operating System updates, see Fig. 1. The content provider is multihomed to three ISPs, and according to the BGP propagation rules, it receives the best BGP route for every destination from each of its provider's routers. The BGP routes installed at each egress router R1 and R2 determine the path the flows will follow to each destination prefix. BGP route selection rules and proper configuration enables a handful of commonly used configurations for the traffic addressed to a given destination:

- Single egress path. The traffic to the destination exits through one of the paths as a result of the BGP route selection rules [35]. For example, the configuration of a higher BGP `LOCAL_PREF` attribute for a path results in all the routers of the AS preferring it. Besides, some of the rules applied to the route attributes received may also result, without additional configuration, in the same route being selected by all routers. For example, if no preference is configured, a route traversing less intermediate networks than the rest of the routes is selected by all nodes. In Fig. 1a, the administrator of the content provider can set a higher priority for the routes received from R3.
- Traffic sharing. Egress routers R1 and R2 may both select the route through their directly connected external router if the routes received from the providers are similar enough[1]. In this case, as depicted in Fig. 1b, traffic addressed to the destination network arriving to router R1 could go through ISP1 (if preferred over ISP2), and traffic arriving to router R2 would go through ISP3, eg., through R5. The amount of traffic egressing through each path depends on the internal systems selecting R1 or R2.
- Equal Cost Multipath. Finally, provided that *Multipath BGP* [9] [23] is enabled in R2, as shown in Fig. 1c, R2 may select both routes advertised by R5 and R6, if they share most of the BGP route attributes, including the networks in the path to the destination (in this case, ISP3). R2 assigns flows to egressing paths with even probability.

After this analysis, we can conclude that the assignment of traffic to paths is:

[1] Same BGP `LOCAL_PREF` values, number of traversed networks, origin and `MED` metric.

*Corresponding author
`pedro.torres@ufpr.br` (P. R. Torres)
ORCID(s): `0000-0003-3793-2840` (P. R. Torres)

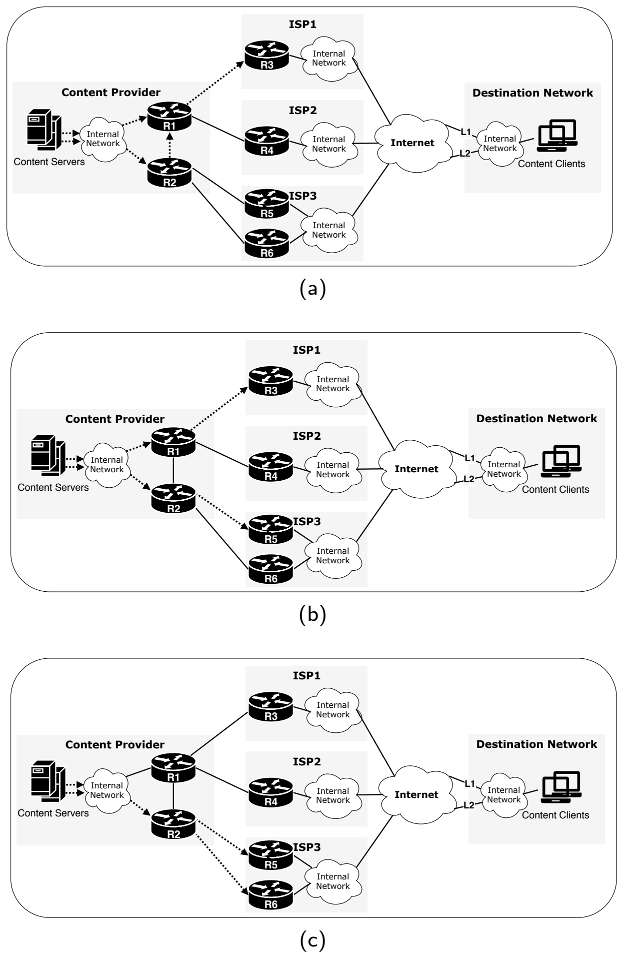


Figure 1: Example of a Content Provider connected to three ISP for (a) single egress path, (b) traffic sharing and (c) multipath BGP. Arrows indicate the egress paths in use.

- **Coarse-grained**. The ability of the current solutions to split egress traffic into the available paths is limited both in the subset of paths and in the ability to flexibly assign traffic to paths. As shown before, standard configurations do not allow a single router to use at the same time two different providers and routers in an AS are restricted to select paths equivalent in `LOCAL_PREF` values, number of traversed networks, origin and `MED` metric. Besides, every node in the content provider can only egress through one network path, even though other nodes could follow a different one.

- **Independent of current path conditions**. The selection of the paths depends on local configuration or on the BGP attributes of the routes received, not in the actual conditions of the paths. For example, routers at the content provider cannot be aware that the available capacity of the link L1 connecting to the destination network in the figure is much lower than L2, for example, or experiencing higher traffic demand.

These problems are a concern for large traffic generators such as Facebook [39] or Google [42].

In this paper we propose Bartolomeu[2], a solution to enable content generating networks to perform adaptive load balancing across multiple interdomain paths by spreading the traffic across available paths according to a passive measurement of path performance. The intuition behind Bartolomeu is as follows: Bartolomeu manages relevant (large) flows, and can move them from one path to another, i.e., perform flow rebalancing. The timescale of Bartolomeu's operation is around tens of seconds. Bartolomeu aims to use all the available BGP paths to a destination by allocating a number of relevant flows per path that is proportional to the available capacity of the path to that destination, if there are more relevant flows than paths, or by using the paths with highest capacities, if there are less relevant flows than paths. The available capacity of the paths is determined by measuring the effective rate of the traffic sent to the destination through each path in the egress nodes. Such a proportional flow-to-path assignment results in inter-flow fairness, i.e., aims to provide the same capacity to every flow. In addition, it equalizes the busy time expectation for each path, assuming that all flows have the same expectation for their remaining time - note that Bartolomeu does not know in advance the duration of each flow. Thus, a proportional allocation reduces the number of path changes that will be needed in the future to keep the paths busy. Regarding to flow performance, this strategy reduces the mean sojourn time, i.e., the mean time to flow completion, compared to *Equal Cost Multipath* (ECMP, flows are assigned with equal probability to any of the egress paths) or single path selection. Throughout the paper, we use this metric to evaluate the gains of the system, as it is directly related with the experience of the users of the system.

The implementation of Bartolomeu's flow scheduling consists of an SDN architecture with a centralized application that programs the SDN switches actually forwarding traffic in the borders of the network. Bartolomeu communicates with another SDN application to get an up-to-date BGP feed with route information for all Internet prefixes. In addition, the SDN controller accesses to per-destination and per-path rate measurements provided by the SDN switches.

As the need to manage a very high number of flows may make such a solution unfeasible, Bartolomeu only changes the egressing path for relevant flows. These flows are identified from a sampled trace, obtained from passive traffic sampling tools such as sFlow [34].

To assess the gains of Bartolomeu, we first develop a queuing theory model that allows the comparison of Bartolomeu's flow scheduling with ECMP and single path selection. With this simplified model, which assumes Poisson arrival and service times, and zero path switching time for a flow, we observe that Bartolomeu has lower sojourn time than ECMP when rates are similar, and lower sojourn time than the fastest single path when path rates differ.

[2]The name come from the famous portuguese Bartolomeu Dias, the navigator unveiling new paths by first turning the Cape of Good Hope.

As real traffic may differ largely from a simple Poisson model, we perform several experiments with traffic derived from real network traces. For this purpose, we collected full traces in two different ASes. With these traces, we test our method for detecting large flows, to see that Bartolomeu can manage up to 80% of the AS traffic.

We implemented two prototypes of the system to perform two different experiments. The first is a SDN implementation managing two interdomain paths, exchanging traffic with the flow pattern (flow activation time and amount of data per flow) defined by the capture corresponding to a single destination prefix. In this experiment we observe the impact of the interfering traffic and the effect of path changes to TCP performance. The reduction of the sojourn time measured is around 35%. Bartolomeu's ability to use all available paths to a destination prefix depends on the flow pattern of the managed traffic, i.e., on the number of concurrent relevant flows and flow sizes. In the second experiment we estimate the performance that can be achieved in an AS by modelling a real flow sequence with a fixed-rate multipath network. For this purpose, we developed a discrete-event simulator fed with the flow traces captured in two different content provider networks. We found that Bartolomeu reduces the mean sojourn time for the traffic served by the network, compared to a network using ECMP, to half when path rates differ by a factor of 3, and to a sixth when path rates differ by a factor of 10. When path rates are similar, it behaves similarly to ECMP (with a slight gain of 2 to 5%). We compute the maximum number of per-flow entries and the maximum entry change request rate to show that the resources required fit in with the specifications of the current generation of SDN switches.

Bartolomeu aims to maximize performance in an end-to-end basis using only the local information gathered at the network layer. Thus, its main contribution is the mechanism to keep all available paths to a destination busy by assigning and rebalancing relevant flows in proportion to the measured egress rate, implemented as an SDN architecture. Bartolomeu's novel approach differs from solutions that assume that the bottleneck is at the first hop, and thus rebalance traffic according to the edge link utilization [21, 39]. It is also different from solutions that maximize end-to-end performance with server application performance data [10, 42], with full topology knowledge [2] or with the collaboration of the remote end [15].

The rest of the paper is structured as follows. A detailed description of the mechanism is given in Section 2. The mathematical theory that allows the comparison of traditional load balancing and Bartolomeu's is shown next. Section 4 describes the traffic datasets of two content providers, which are used in the experiments to evaluate Bartolomeu, a simple SDN-based implementation operating over real Internet paths (Section 5) and a discrete-event simulator which operates with the whole traces of each provider (Section 6). The literature background and related work are described in Section 7. We end with the conclusion in Section 8 and the information about the source code and data used in this work in Section 9.

2. Detailed Mechanism

Bartolomeu is an integrated system for a SDN environment that allows the dynamic exploitation of the available path diversity for each BGP destination prefix by adapting the load distribution proportionally to an estimation of the available path capacity. This is achieved by moving egressing flows, unidirectional packet streams with the same source and destination IP addresses, source and destination ports, and protocol, represented respectively by the 5-tuple $\langle sa, da, sp, dp, pr \rangle$.

To make the solution scalable, Bartolomeu only operates on *relevant flows*. We consider a trace sampled with sampling rate S, i.e., on average, one sample will be captured for every S packets observed. We define *relevant flow* as a flow with a duration of at least D, that has been sampled at least s times within an *observation window* of W. Requiring the appearance of at least s (with $s > 1$) samples of a flow within a period W reduces the chance that small flows are selected. Our method for detecting relevant flows follows Mori et al. [32]. In this paper they discuss the probability of false positives (identifying a small flow as an elephant) and false negatives (failing to capture an elephant flow). They find that the threshold value s for a given set of probabilities for false positives and negatives is similar for different distributions of the number of per-flow packets in the unsampled trace, making this method quite robust. Ideally, the number of bytes pending for transmission when these relevant flows are identified should be large, so that reroute operation gains exceed the overheads. This assumption is justified in Section 4.

Bartolomeu can re-assign relevant flows to different egress paths, i.e., to BGP next-hops. In the Introduction we argued that current BGP AS deployments suffer from lack of flexibility in assigning traffic to paths. To enhance the ability of current ASes to split traffic into the available paths we rely on an SDN architecture, in particular, a BGP-SDN architecture (as in [38, 28, 12, 17, 26, 43, 22, 27]), in which BGP routers are replaced partially or totally with switches under central control. As a requirement, we assume that all the switches with external connections are under Bartolomeu's control. These switches, and maybe other switches internal to the network, can be orchestrated to change a route for a given destination prefix or/and be able to install specific rules to change an egress path for any flow. Also, ECMP can be available in the switches to perform (non-weighted) load balancing across available interfaces.

We now describe the four modules that compose the Bartolomeu SDN system (see Fig. 2), the Route Information Module, the Flow Information Module, the Path Information Module and the Load Balancing Decision Module.

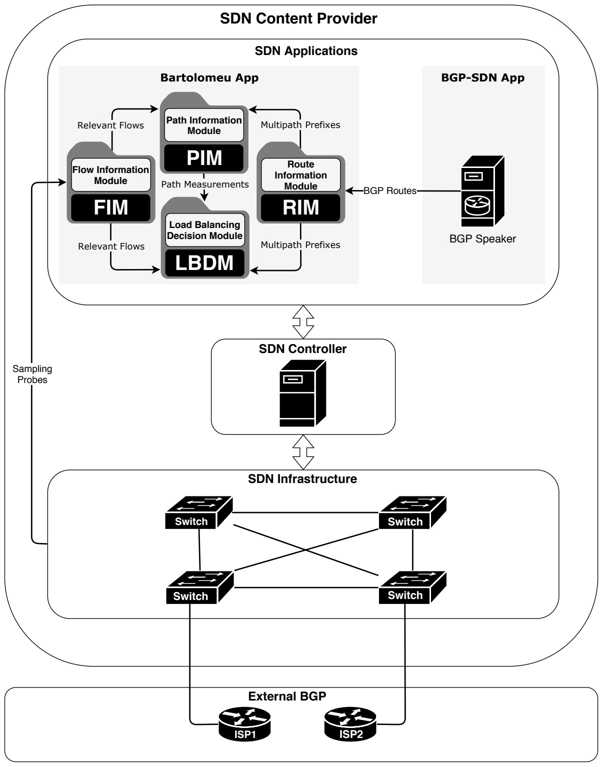


Figure 2: Integration of Bartolomeu modules in a BGP-SDN environment.

2.1. RIM - Route Information Module

RIM is responsible for gathering all the routes available at the local AS for any destination. It obtains the routing information from external BGP peers and returns a report with the next-hops for each multipath prefix. Note that RIM provides a unified view of the routes that can be used by the AS, regardless of the actual egress switch to which they correspond.

To efficiently gather routes from many BGP feeds, RIM can be split into different software instances, as proposed for OFBGP [12]. OFBGP is a BGP-SDN system that distributes the centralized BGP processing into different execution task units. Different execution task units can be deployed to gather routes from different peers, and once the BGP protocol data has been retrieved, the BGP route selection process can be split into different execution task units according to the destination prefix. In addition, the RIPE collector infrastructure also describes a similar architecture that is able to gather full BGP route tables from hundreds of routers with reaction times in the order of seconds [36]. Bartolomeu can use a modular architecture similar to those described above to access to and process BGP route information timely in order to assist load balancing decisions.

2.2. FIM - Flow Information Module

FIM is the module in charge of identifying the relevant flows. It acts as a collector that receives the sFlow [34] probes sent by the agents embedded in the network devices and subsequently processes them in order to identify flows that have potential to be moved from one path to another. We assume the SDN switches at the AS can be configured to sample traffic at a given rate S per egress interface. The information received from all switches using sFlow is processed for every *observation window* W. Then, FIM selects flows with at least s samples in the *observation window*, with a maximum separation of the samples of a flow of at least D seconds. We discuss the rationale and suggested values for these parameters in Section 4.

2.3. PIM - Path Information Module

PIM provides a measure of the rate per egress path and per destination. PIM uses FIM and RIM information to identify ⟨BGP prefix, next-hop⟩ pairs for which a relevant flow is assigned. For each ⟨BGP prefix, next-hop⟩ pair, PIM issues a monitoring rule request to the SDN controller to trigger bitrate monitoring in the corresponding switch. Only ⟨BGP prefix, next-hop⟩ pairs for which a relevant flow is assigned are monitorized. In this way we try to assure that we only measure paths in which at least one flow aims to use greedily the bandwidth available at the path.

With period of quantum q, PIM requests the SDN controller to obtain the amount of bytes transmitted for the ⟨BGP prefix, next-hop⟩ pairs with at least one flow. To smooth the measured value, we compute the *Exponential Moving Average* (EMA) of the bytes transmitted in the last periods, with α as weight decrease factor.[3] If the rates of the traffic through next-hop h for a given prefix, B_h, do not differ significantly from the observation at the previous period count, B_h^-, i.e., $|\frac{B_h - B_h^-}{B_h}| < \epsilon$, we set $B_h = B_h^-$. The parameter ϵ acts like a dumping factor where small changes in the throughput of the paths are ignored in the input to the REBALANCE procedure of the LBDM module, described next.

2.4. LBDM - Load Balancing Decision Module

Bartolomeu aims to keep all available paths to a given destination busy with the minimum number of flow redirections, as path changes may result in delays and retransmissions. To achieve this goal, Bartolomeu detects the active flows that are relevant, i.e., large enough to be managed by the traffic control system, and the rates measured through different paths to their corresponding BGP destination. We note that Bartolomeu, that operates at the network layer, is not able to estimate the remaining size of each flow. Thus, Bartolomeu assumes that all active flows have the same amount of data to transmit, so proportional traffic distribution to the measured egress rate implies proportional distribution of the number of flows. As inter-flow throughput fairness results from this strategy under the hypothesis of equal remaining size, the expected sojourn time is equalized for the active flows, and the number of paths changes needed to keep all paths busy according to these hypothesis would be minimal.

[3]The value of α can be adjusted between 0 and 1 to control the influence of older observations.

LBDM is the module in charge of moving relevant flows from one egress path to another so that the number of flows assigned to each path is proportional to its measured rate. However, LBDM has to fulfill an additional condition: each next-hop must have at least one flow if possible, in order to use all available paths and ensure fresh rate measurements for each of them. Thus, we first assign one flow to each path, and then proceed to the proportional distribution of the remaining flows. A particular case occurs if the number of flows to assign is lower than the number of next-hops available for the prefix; in this case, the paths with higher measured rate should receive one.

Let $F = \{F_1, ..., F_m\}$ be the number of active flows to a given BGP prefix destination, for m different next-hops, and $\sum F$ the total number of active flows for that destination. $B = \{B_1, ..., B_m\}$ are the measured rates per egress path to the destination, and $\sum B$ its sum. The case in which $\sum F \leq m$ has the trivial formulation of assigning one flow to the egress paths ordered in descending rate order. For the case $\sum F > m$, we can formalize the problem as follows:

$$\begin{aligned}
&\text{find} && F^+ = \{F_1^+, ..., F_m^+\} \\
&\text{with} && F_h^+ \approx 1 + [(\textstyle\sum F) - m]\frac{B_h}{\sum B} \\
& && \forall\, h \in [1, ..., m] \\
&\text{subject to} && F_h^+ \geq 1, \\
& && \textstyle\sum F^+ = \sum F, \\
& && F_h^+ \in \mathbb{Z}.
\end{aligned}$$

Next, with the current allocation F and the target allocation F^+, LBDM has to determine the flows to move with the objective of minimizing the number of flow changes.

In the next paragraphs we describe REBALANCE, that solves the problem stated above. The algorithm is shown in Fig. 3. It is executed for each prefix every time new rate information is obtained, i.e., every q seconds.

REBALANCE first triggers the execution of ComputeFlowCountTarget to determine F^+, the number of flows assigned to each next-hop h, out of the total active relevant flows. If the number of flows to assign is lower than the number of next-hops available for the prefix, the paths with highest measured rate receive one. Otherwise (i.e., $\sum F > m$), it assigns one flow to each egress path, so that $F_h \geq 1$. To approximate to the distribution of the additional flows in proportion to B_h, each path is first assigned the floor of its share on the remaining flows, $\lfloor(\sum F - m)\frac{B_h}{\sum B}\rfloor$. After this process, there may remain some flows, less than m, to distribute among the egress paths. We have chosen the *Largest Remainder Method* (LRM) [16] as the tie-break criteria to distribute them.

Once the number of flows each egress path should have is known, flows are moved from one egress path to the other, with the objective of moving the fewest number of flows possible. The procedure ComputeBalanceAndReassignFlows identifies egress paths with more and less flows than required, so that flows can be moved from the first set of paths to the second. The MoveFlows function selects the particular flows to move from the path with excess. This function selects the oldest flows in an egress path among those with a minimum number of previous path changes.

ComputeFlowCountTarget's complexity is determined by the two ordering operations performed with the next-hop list, accounting for $O(mlog(m))$. The ComputeBalanceAndReassignFlows procedure iterates over the list of next-hops to match paths with excess of flows with paths with deficit, and then performs a flow selection which depends on a sorting procedure of the active flows active on each path. Thus, the complexity can be expressed as $O(m^2 + F_h log(F_h))$. Overall, the complexity of the second procedure dominates.

We have discussed how to rebalance traffic by moving relevant flows. However, egress traffic performance also depends on the assignment of new flows to egress paths. As described in the introduction, current ASes use either a single route to a destination prefix (BGP best path selection) or distribute traffic equally among several paths (BGP Multipath), in both cases independently of the current route conditions. LBDM can leverage the information provided by RIM, FIM and PIM to improve default path selection. We present two strategies that can be implemented with the forwarding strategies available in common routers. The first is to set the egress path with highest measured rate B_h as the next-hop for each flow to a given BGP destination. We call this strategy Bfast, Bartolomeu's fastest measured path. The second is to determine the weights of a *Weighted Cost Multipath* (WCMP) [44] system following the B_h distribution, if the hardware supports WCMP. We note that WCMP cannot be combined with the REBALANCE procedure, as the change in the weights resulting from WCMP's weight adjustment results in egress path changes to existing flows. In this case, both WCMP and REBALANCE interfere with each other, pursuing the same objectives by different means. On the other hand, the REBALANCE procedure can be safely combined with BGP best path, ECMP and Bfast as default egress path allocation. In Sections 5 and 6 we compare several combinations of these techniques for default path selection and REBALANCE (e.g., ECMP and no REBALANCE, WCMP, ECMP with REBALANCE, etc.)

We finally note that Bartolomeu should operate in a larger timescale than the control timescale of TCP, so that they do not compete with each other. This goal can be achieved by proper selection of q, that, as previously stated, drives the execution of the REBALANCE procedure. Gao et al. [19] suggest that a minimum period for TCP flow reallocation should be in the order of few tens of seconds.

3. Mathematical Model

In this section we provide a simple model for estimating the mean time a flow spends in a system performing the REBALANCE process described for Bartolomeu, i.e.,

Algorithm 1

Parameter Description:
H: List of next-hops for the prefix p (provided by RIM).
F: List of the number of relevant flows for each next-hop in H to prefix p (provided by FIM).
B: List of the measured bitrate of the traffic to prefix p, for each next-hop in H (provided by PIM).
F^+: List with the target number of flows to assign to each next-hop, result of ComputeFlowCountTarget.

Functions Used:
NextHopListOrdered(H, C_1, C_2): Receives a list of next-hops identifiers, and two lists with a value for each next-hop, C_1 and C_2. Returns a list of next-hops ordered according to the values of C_1, and for equal values of C_1, ordered according to C_2.
MoveFlows($Amount, Source, Destination$): Receives a number of flows to move, the source and destination next-hops and performs the reassignment of flows.

```
procedure REBALANCE($H, F, B$)
    $F^+ \leftarrow$ ComputeFlowCountTarget($H, F, B$)
    ComputeBalanceAndReassignFlows($H, F, F^+$)

procedure ComputeFlowCountTarget($H, F, B$)
    $Bsum \leftarrow 0$
    $FlowsToAssign \leftarrow 0$
    $H \leftarrow$ NextHopListOrdered($H, B, \emptyset$)    ▷ Next-hop list ordered according to $B$, top rated paths receive a flow when $f < m$.
    for each $h \in H$ do
        $FlowsToAssign \leftarrow FlowsToAssign + F_h$    ▷ Total of flows in $F$
        $Bsum \leftarrow Bsum + B_h$    ▷ Sum of bitrates in $B$
        $R_h \leftarrow 0$    ▷ Initialize the remainder $R$ with zero
    for each $h \in H$ do
        if $FlowsToAssign \geq 1$ then    ▷ Try to give one flow per next-hop
            $F_h^+ \leftarrow 1$
            $FlowsToAssign \leftarrow FlowsToAssign - 1$
        else
            $F_h^+ \leftarrow 0$
    if $FlowsToAssign \geq 1$ then    ▷ Apply Largest Remainder Method
        $Quote \leftarrow Bsum \ / \ FlowsToAssign$
        for each $h \in H$ do
            $BitratePerQuote \leftarrow B_h \ / \ Quote$
            $Floor \leftarrow \lfloor BitratePerQuote \rfloor$
            $R_h \leftarrow BitratePerQuote - Floor$    ▷ Update remainder value for this next-hop
            $F_h^+ \leftarrow F_h^+ + Floor$    ▷ Integer-part flow allocation
            $FlowsToAssign \leftarrow FlowsToAssign - Floor$
    $H^{'} \leftarrow$ NextHopListOrdered($H, R, F^+$)    ▷ Next-hop list ordered according to $R$ (and $F^+$ for equal $R$)
    for each $h \in H^{'}$ do
        if $FlowsToAssign \geq 1$ then
            $F_h^+ \leftarrow F_h^+ + 1$    ▷ Next-hop $h$ receives unallocated flow
        $FlowsToAssign \leftarrow FlowsToAssign - 1$
    return $F^+$    ▷ There is the same number of flows in $F$ and $F^+$

procedure ComputeBalanceAndReassignFlows($H, F, F^+$)
    for each $h \in H$ do
        $\Delta_h \leftarrow F_h^+ - F_h$    ▷ Compute the balance between target and existing allocations
    for each $d \in H$ do
        for each $s \in H$ do
            if $\Delta_d > 0$ and $\Delta_s < 0$ then    ▷ Move flow from next-hop $s$ (source) to $d$ (destination)
                $ToMove \leftarrow$ min($\Delta_d, |\Delta_s|$)
                MoveFlows($ToMove, s, d$)
                $\Delta_d \leftarrow \Delta_d - ToMove$
                $\Delta_s \leftarrow \Delta_s + ToMove$
```

Figure 3: Rebalancing Algorithm for the prefix p with Largest Remainder Method.

the mean response time or mean sojourn time. The aim of the model is to provide a useful abstraction to reason about the mechanisms and the tradeoffs involved in its design. Besides, it allows a fast-to-compute comparison of REBALANCE with state-of-the-art flow assignment techniques such as ECMP splitting or the use of single route per destination. The results obtained indicate that REBALANCE performs at least as well as the best of the alternatives considered.

We first describe a queuing model that represents the transmission of several bulk transfers over a path for the three options considered (REBALANCE, ECMP, fastest single path). Then, we develop a Markov chain to derive the mean sojourn time for exponential arrival and service time distributions. Finally, we compare the three flow assignment policies and derive some conclusions.

The main limitation of the analysis lies in the mismatch of the characteristics of real traffic and the Poisson model we use to obtain simple expressions for the mean sojourn time. For example, among the studies which model the *inter-arrival* time for sessions, flows and packets, Arfeen et al. [4] suggest a possible fit with the Weibull distribution, and Downey [11] reports some evidence that the distribution of TCP flow *transfer times* can be long-tailed. Thus, the results of this section are only intended to sketch the performance trends resulting from each mechanism, instead of allowing an accurate performance estimation. The reader is referred to Sections 5 and 6 for results obtained with a realistic distribution of arrival and service times.

3.1. FIFO realization of REBALANCE, ECMP and fastest single path

To compute the sojourn time for several bulk transfers over a path, we first assume that the active flows share an equal fraction of the available capacity. A similar problem, known as *Processor Sharing* (PS), was studied by Kleinrock to represent an idealized round-robin processor scheduling [25]. The PS queuing model has been used before to analyse the flow completion time (service time) for TCP flows [30, 33]. Although the problem of sharing a resource, a server, according to the PS model differs from a classical FIFO system in which individual customers occupy the server until the job finishes, while the remaining customers waits in a queue, both PS and FIFO formulations share some performance figures: The number of flows in the PS system is the same as the number of customers in its equivalent FIFO queuing system [25]. According to Little's formula, the mean sojourn time in steady state for a PS system is the same as that of a FIFO system with the same arrival, number of servers and service rates. Thus, in the next paragraphs we compute the mean sojourn time for a FIFO realization of REBALANCE, ECMP and fastest single path, as the results coincide with the PS formulation, and thus correspond to the TCP service time problem.

Lets consider an AS with m different egress paths to a given prefix, each one with a different service bitrate, B_h, ordered from highest to lowest rate, and f active flows. The FIFO realization of Bartolomeu's REBALANCE is defined as follows: when $f > m$, all egress paths are occupied with a single flow, and the remaining flows wait in a queue to be serviced, according to the FIFO assumption. When $f \leq m$, one flow is assigned to each of the first f egress paths in the rate order (from highest to lowest rate). In this case, when a flow at link $i < f$ finishes, the flow in the path with lowest capacity (in path position f) is rescheduled to path i to ensure that the paths with highest capacity are always in use. This model does not take into account several time parameters defined for Bartolomeu operation, such as the flow observation window W or the quantum q between rate measures and rebalance trigger, which are assumed to be zero.

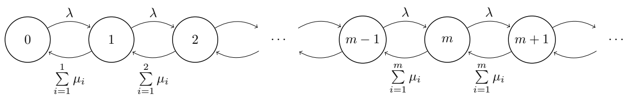


Figure 4: State-transition-rate diagram for $M/M/\vec{m}$.

We now present the ECMP FIFO realization: ECMP aims to spread evenly the traffic to the same destination by assigning each flow to one of multiple available paths according to the hash performed with the parameters identifying the flow. In this case, however, flows are not rescheduled to empty paths with higher capacity. The flow-to-path assignment is therefore modelled as the result of random choice, with probability $\frac{1}{m}$.

Finally, the choice of a single path is easy to model: all flows are assigned to a single server, so that all $f-1$ flows keep waiting.

3.2. Poisson model for FIFO realization of flow assignment policies

We now compute the sojourn time for each flow assignment police. We use a Poisson model to represent arrival ($\lambda > 0$) and service times (μ_h for egress path h, $\mu_h \geq \mu_{h+1} \forall h$), in order to be able to compute the mean sojourn time. We extend Kendal's notation to call $M/M/\vec{m}$ the Poisson FIFO model for Bartolomeu's REBALANCE, where $\vec{m}$ stands for m egress paths with different service times. We present how to compute the mean response time (sojourn time) for $M/M/\vec{m}$.

The state-transition-rate diagram in the $M/M/\vec{m}$ case is shown in Fig. 4. This diagram represents a continuous-time Markov process in which the probability of changing to a state with one more flow in the system depends on the arrival rate, and the probability to reduce one flow depends on the sum of the service rate of all the servers active (in descending order of service capacity). Such a Markov process is known as a *birth-death* process. In steady-state, the rate at which flows change from state i to $i+1$ is the same as in the opposite direction, from $i+1$ to i. Thus, $P_0\lambda = P_1\mu_1$, $P_1 = \frac{P_0\lambda}{\mu}$. By induction, we compute the probability P_n for the n^{th} state, $n = 1, 2, ..., m$:

$$P_n = \begin{cases} \frac{\lambda^n}{(\mu_1)(\mu_1+\mu_2)...(\mu_1+...+\mu_n)} \times P_0 & \text{for } n \leq m, \\ \frac{\lambda^n}{(\mu_1)(\mu_1+\mu_2)...(\mu_1+...+\mu_m)^{n-m}} \times P_0 & \text{for } n > m. \end{cases} \quad (1)$$

As $\sum_{n=0}^{\infty} P_n = 1$, we can solve the equations (with M the sum of all μ_i capacities, $M = \sum_{i=1}^{m} \mu_i$)

$$P_0 = \frac{1}{1 + \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1(\mu_1+\mu_2)} + ... + \frac{\lambda^m}{Y} + \sum_{i=m+1}^{\infty} \frac{\lambda^i}{YM^i}} \quad (2)$$

where $Y = \mu_1(\mu_1+\mu_2)...M$.

We can replace the geometric infinite power-series in right side of the denominator at (2) for its closed-form, provided that $\frac{\lambda}{M} < 1$, to obtain:

$$P_0 = \frac{1}{1 + \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1(\mu_1+\mu_2)} + ... + \frac{\lambda^m}{Y} + \frac{\gamma}{1-\eta}} \tag{3}$$

where $\eta = \frac{\lambda}{M}$ and $\gamma = \frac{\lambda^{m+1}}{YM^{m+1}}$.

With P_0 we are able to calculate any other probability. The probability that an arriving job is forced to wait in the queue is given by:

$$P\{Q\} = 1 - \sum_{i=0}^{m} P_i \tag{4}$$

With the queuing probability, the average time jobs spend in the queue can be calculated as:

$$E[W] = \frac{P\{Q\}}{M - \lambda} \tag{5}$$

Thus, finally, can obtain the average time jobs spend in the system, or the *sojourn time* as:

$$E[R] = E[W] + \frac{1}{\hat{\mu}} \tag{6}$$

where $\hat{\mu} = \frac{\mu_1 P_1 + \mu_2 P_2 + ... + \mu_m P_m}{\sum_{i=1}^{m} P_i}$.

We now compute the response time when the classical ECMP load-balancing is applied. ECMP spreads flows to different egress paths with even probability (regardless the capacity of the path), problem also known as *Bernoulli splitting* [14]. If we consider m egress paths, each one with capacity μ_i, $(i = 1, 2, ..., m)$, each of the paths receive new flows with rate $\frac{\lambda}{m}$ (each egress path is equiprobable). The sojourn time can be expressed as the mean of m independent $M/M/1$ queues:

$$E[R] = \sum_{i=1}^{m} \frac{1}{m}(W_i + \frac{1}{\mu_i}) = \frac{1}{m}\sum_{i=1}^{m}(\frac{1}{\mu_i - \lambda_i} + \frac{1}{\mu_i}) \tag{7}$$

Finally, for the single path flow assignment to path i, the $M/M/1$ model with $\mu = \mu_i$ applies.

3.3. Comparison between flow assignment policies

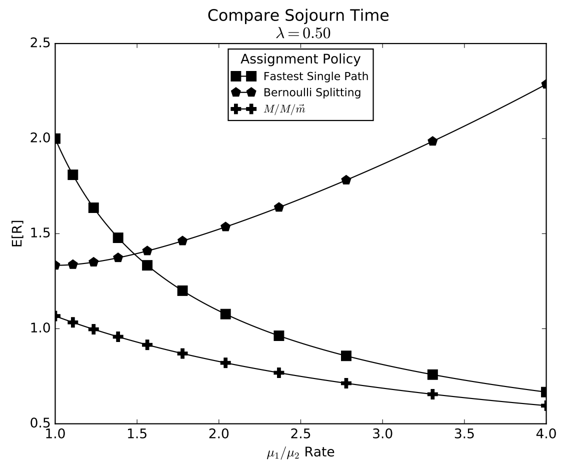
We compute the sojourn time with the expressions presented in the previous subsection, in particular according to Equations 6, 7 and $M/M/1$, respectively, for a topology with two egress paths, and show the results in Fig. 5. We represent the sojourn time for $M/M/\vec{m}$ (REBALANCE), Bernoulli (ECMP), and the selection of the fastest egress path. Note that the selection of the fastest egress path would require measuring the best path for every destination, so it represents an upper bound of the results that can be obtained by a single path assignment strategy. Without measurement, the slowest path could be selected as the default route for this prefix and the sojourn time would degrade.

For the analysis, we select as arrival rates $\lambda = 0.50$ and $\lambda = 0.95$, respectively. Several service times μ_1 and μ_2 were used, with $\mu_1 \times \mu_2 = 1$. As expected, ECMP performs better than fast path when path capacities are similar, and fast path performs better than ECMP when there are large differences. $M/M/\vec{m}$ performs better than both of them. As the rate differences grow, fastest path and REBALANCE are increasingly similar – the difference between them is the ability of REBALANCE to move a flow to the fastest link when the flow previously allocated finished and there are no more flows in the queue. When the rates of the two paths are similar, ECMP and REBALANCE approximate. Lower arrival rates ($\lambda = 0.50$) increase the probability that a low number of flows are in the system, and in this case REBALANCE reduces the main sojourn time by moving flows that were initially allocated to the same link to a free one.

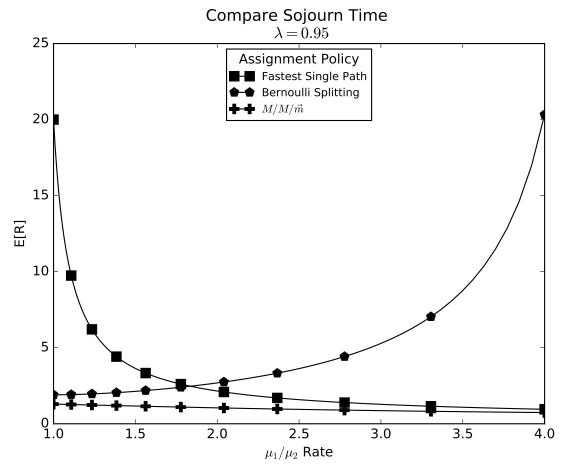
4. Flow dataset description and analysis

As described in Section 2, Bartolomeu uses the FIM module to identify the flows to move. Therefore, Bartolomeu's performance heavily depends on the characteristics of the flows to which it is applied, such as the share of the total traffic that correspond to the flows that can be reassigned, and the remaining duration of a reassigned flow. The larger the number of flows that are identified by the FIM and the larger aggregate traffic share associated to these flows, the better the performance. However, the number of flows to identify is limited by the sampling rate the inspection infrastructure can achieve, its ability to transmit this information to the FIM module, and the time to process it. Besides, the LBDM module may not be able to use all the flows identified by FIM. SDN hardware may impose some limits both in the number of flows to manage at a given time and the number of table updates per second. In this case, the FIM module should be configured to select the most relevant flows that fit in the restrictions imposed by the infrastructure. In this section we analyze the traffic traces of two content providers to present some guidelines for the values of the FIM parameters to ensure that it operates within reasonable hardware limits, and at the same time, allow Bartolomeu to improve performance.

We accessed two 1-hour long set of traces of networks serving large content files. The RNP (Rede Nacional de Ensino e Pesquisa, the Brazilian academic network) dataset was collected at a Point of Presence (PoP) in Curitiba/Brazil [37] and the WIDE dataset, collected on the backbone connection between WIDE and DIX-IE, an Internet eXchange Point (IXP) in Tokyo/Japan [8]. These whole-traffic captures correspond to the egress direction and contain unmodified layer 3 and 4 headers. IP addresses are matched with the BGP information of a router at the AS at the time the traces were captured. Table 1 shows a summary of the traces. Despite the similarity in relation to throughput, we highlight that the



(a)



(b)

Figure 5: Sojourn time for $M/M/\vec{m}$, multipath Bernoulli, and fastest single path, computed according to Equations 6, 7 and $M/M/1$, respectively. The service rate for the two egress paths are μ_1 and μ_2, with $\mu_1 \times \mu_2 = 1$. The graph (a) considers an arrival intensity $\lambda = 0.5$ and graph (b) $\lambda = 0.95$.

Source	Location	Date	Mbps	GB	Flows
RNP	Curitiba	2018-05-04	3393	1445	7,287,284
WIDE	Tokyo	2018-05-09	2964	1242	20,063,499

Table 1
Summary for RNP and WIDE traces

number of 5-tuple flows observed across the 1-hour period varies greatly.

Bartolomeu's FIM defines a *relevant flow* as a flow with at least s samples within an observation window of W, with a duration (largest difference between two samples in an observation window) of more than D seconds. In this way, we aim to remove short-lived flows, with duration less than D. The use of high-enough values for S and s in a short observation window W, enables to select only flows with a high rate during this period, i.e., it eliminates flows with fewer data to transmit, as its reallocation would be worthless. Besides, the traffic monitoring hardware may impose a further limit in the sampling rate that can be supported.

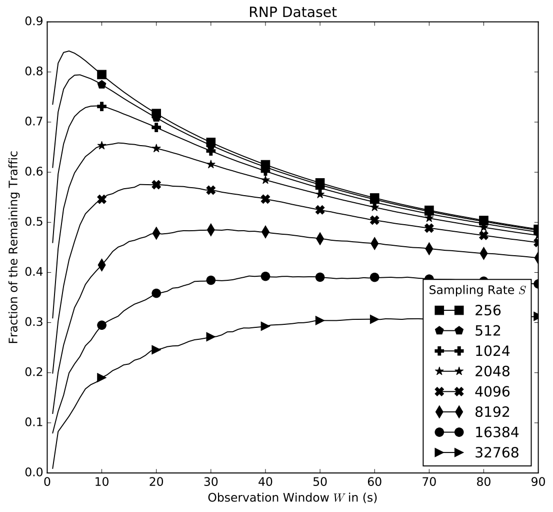
We now present the analysis of the dataset traces according to the FIM processing rules. For this analysis we have set s to 2 (i.e., at least two packets must be sampled in an observation window W), and D to $W/2$ (there must be two samples in the observation window with a time difference larger than $W/2$). Fig. 6 shows the fraction of the sum of the traffic corresponding to the flows identified by FIM for different values of the observation window W (x axis) and of the sampling rate S (different curves), for both RNP and WIDE datasets. We only count the amount of bytes observed in the trace after the flow selection process has been performed, the remaining traffic, which are the bytes that could be moved by Bartolomeu to a different egress path. The higher the y-axis value, the larger the amount of traffic managed by Bartolomeu, and the higher the expected gain. We

Dataset	S	W (s)	Remaining Traffic (%)	# TCP Flows (in 1 hour)	# BGP Prefixes
RNP	256	4	84.19	87987	11302
	512	6	79.43	59010	9370
	1024	9	73.23	38807	7408
	2048	13	65.84	24442	5523
	4096	17	57.56	15010	4082
	8192	33	48.55	8871	2897
	16384	37	39.29	4967	2037
	32768	83	31.39	2115	1153
WIDE	256	3	80.05	78518	3695
	512	5	76.56	49065	3179
	1024	8	72.18	30152	2624
	2048	10	66.39	17757	2049
	4096	12	60.19	10455	1632
	8192	39	53.85	5138	1183
	16384	43	48.77	2707	829
	32768	90	43.39	1420	529

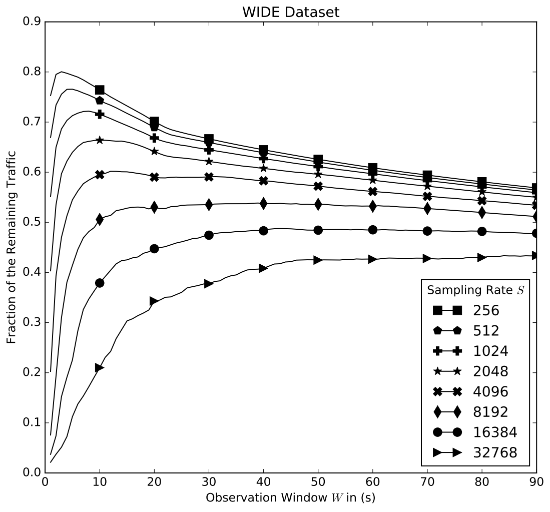
Table 2
Summary for optimal values of S and W to select most of the remaining traffic.

confirm with the graphs that the selected relevant flows can account for a large fraction of traffic. With sampling ratios of more than 1:2048, we can manage more than 50% of the traffic. For these cases, the optimal observation window is below 20 s. These observations hold for both datasets.

Table 2 represents measured values of different parameters for the optimal observation window for different sampling rates. We observe that the amount of pending traffic corresponding to relevant flows, after their identification as relevant flows, i.e., the traffic that can be managed by Bartolomeu, can exceed 80%. Besides, this traffic is aggregated in less than 100,000 flows (out of the 7 to 20 million of flows observed in the same period, see Table 1) and with less than 11,500 prefixes to monitor, which represents only a small fraction (about 1.5%) of the BGP routing table at the time of this analysis.



(a)



(b)

Figure 6: Fraction of the remaining traffic corresponding to the flows identified by the FIM module for many Sampling Rate (S) and Observation Window (W), using the minimum duration $D = \frac{W}{2}$ and number of samples $s \geq 2$. The graph (a) correspond to RNP dataset and the graph (b) to WIDE dataset.

5. Experimental analysis of Bartolomeu over Internet paths

To evaluate Bartolomeu, we first perform an experiment to download several files across different Internet paths. The files correspond to a sequence of flows for a single destination. The objective of this experiment is to observe in the measured sojourn time the effect of real background traffic and the impact in TCP of the path changes induced by traffic rebalance.

For this experiment we developed the PIM and LBDM modules as an in-built application running on the SDN controller, based on the functionality provided by the Ryu framework. Note that the services provided by the RIM and FIM modules were not necessary for this experiment, as the routes for the single destination prefix are manually configured over the Internet, and the flow activation pattern is extracted from real traces. The controller interfaces an Open vSwitch virtual device, communicating through OpenFlow 1.3. This implementation is restricted to manage a single BGP prefix, with multiple paths to the destination. The controller periodically polls the switch to obtain the amount of egress data corresponding to the destination prefix, according to PIM functionality. With the measured per-path rate, the controller computes the target for the number of flow per path, decides which flows to move, and configures the switch accordingly. The SDN controller removes entries for flows inactive for more than 20 s. Regarding the identification of the flows, we assume for experimental purposes that all TCP flows are relevant.

In addition to Bartolomeu's PIM and LBDM, the controller also takes care of the initial flow-to-path allocation, implementing ECMP, WCMP or fast path policies. In a real system, this allocation could be performed by the switch itself, for example, by configuring ECMP in a Multipath BGP capable router.

In the network scenario deployed, see Fig. 7, two VPNs connect locations in Madrid (Spain) and Curitiba (Brazil). VPN_1 follows a path P_1 across [RedIris, Geant, RedClara, RNP] and the other, VPN_2 uses a path P_2 across [RedIris, GTT, NTT, RNP]. These VPNs are established over the common Internet infrastructure, so they are subject to traffic interference. Each path has distinct bottlenecks, P_1 with a mean rate of 25 Mbps during the experiment, and P_2 of 70 Mbps. The measured mean RTTs are 240 and 251 ms, respectively.

TCP connections are established between Linux hosts with default cubic congestion control. Flow initial times and sizes correspond to one of the destination prefixes of the RNP trace. Following a real trace for a prefix from RNP, we define a flow activation schedule in which 48 TCP flows are generated. The total amount of data exchanged is 15.72 GB, going from Madrid, the place at which Bartolomeu is deployed, to Curitiba.

For these experiments, we selected the following values for the parameters that determine PIM and LBDM operation, $\alpha = 0.70$, $\epsilon = 5\%$ and $q = 20$. The experiment duration depends on the way flows are allocated to paths, and ranges from 25 to 41 minutes.

This configuration allows us to measure the sojourn time of the flows with real background traffic. We evaluate the gain that Bartolomeu can provide considering different combinations of initial flow assignment methods and the REBALANCE procedure.

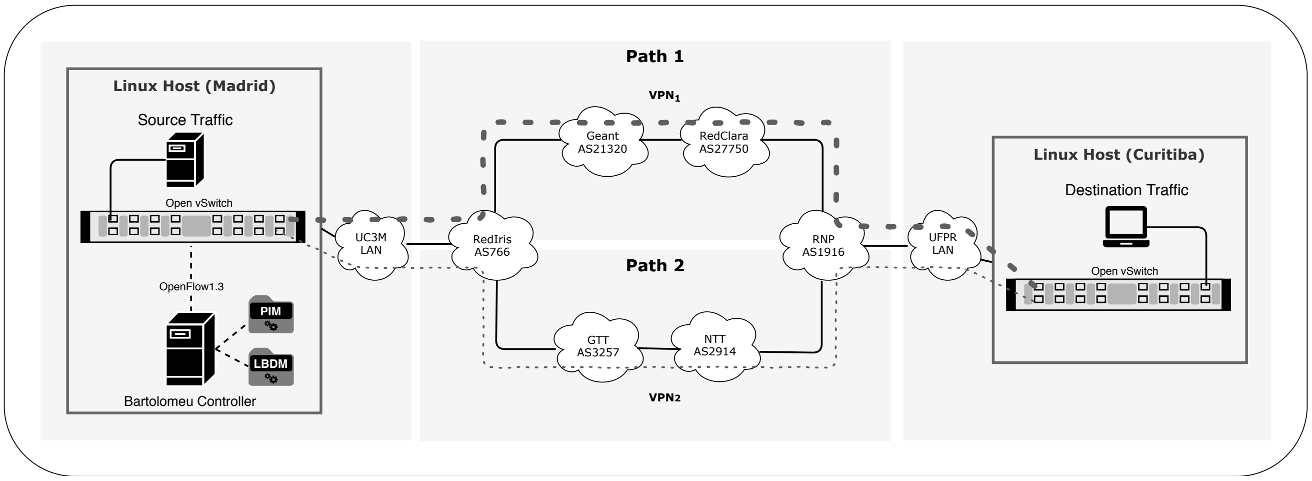


Figure 7: Scenario used in the experiment over the Internet paths.

To perform initial flow allocation we use ECMP, WCMP and Bfast. For WCMP and Bfast, traffic is split accordingly to the rate measured by PIM. Note that the results presented for WCMP can only be obtained if a system similar to Bartolomeu provides the measured rates for each destination prefix. In addition, Bfast only makes sense when PIM and REBALANCE is active, as PIM measures and REBALANCE ensures that one flow is allocated to each path different to the current fastest one, to provide measures of all egress paths. For ECMP, we run the experiments considering the REBALANCE procedure either on and off.

To provide insight on the behavior of Bartolomeu, we now focus on one particular experiment. Fig. 8 shows the number of flows assigned to each egress path according to different strategies. The subgraph at the bottom depicts the starting time and size (y axis) of the flows used for the experiment. The communication is mainly bursty, with all flows started in less than 200 s. ECMP without REBALANCE represents a default configuration of a non-Bartolomeu system. In this configuration, flows are equally split on both paths, with a difference in the time each path completes its assignment proportional to its mean rates (70 and 25 Mbps). WCMP (second subgraph) improves the distribution of the flows once a measure of the rates is available. To illustrate the gains rebalance provides, we observe that in this particular experiment the slower path has been assigned with larger flows - note that the flow assignment procedure is unaware of the size of the flows to allocate. Therefore, the time required to complete the paths following the slower path is larger. When REBALANCE is active (third subgraph), remaining flows can be moved to the most appropriate path. We observe that the finishing time for the last flow is similar to all the cases in which rebalance is on.

In Fig. 9 we observe the number of moved flows for the previous experiment, ECMP and Bfast with REBALANCE. This number is higher when the initial allocation does not consider the rates of the paths. The number of moved flows during the experiment duration is in all cases lower than the total number of flows (48), showing that the mechanism does not incur in excessive churn.

Initial Assignment	Rebalance (on/off)	Mean Sojourn (s)	Flows Moved (max)
ECMP	off	672.63	-
WCMP	off	573.23	-
ECMP	on	495.24	33
Bfast	on	490.34	26

Table 3
Results of Bartolomeu SDN implementation with Internet paths.

We now present the results for 20 repetitions of each experiment. The results are shown in Table 3, and discussed below:

- ECMP with no REBALANCE, which represents non-Bartolomeu operation, experiences the larger sojourn times.
- WCMP provides an improvement of 17% over ECMP.
- ECMP with REBALANCE and Bfast with REBALANCE achieve a 35% to 37% reduction in the sojourn time, respectively. This improvement appears despite the performance costs path changes cause to TCP, which affects (in the worst case) to more than half of the flows.

We observe in this experiment that flow rebalance has more impact in the sojourn time than the initial flow allocation strategy. These results obtained in an scenario with real bottlenecks support Bartolomeu's main hypothesis that the use of the rate measured per egress path to influence the egress path selection results in reductions of the sojourn time.

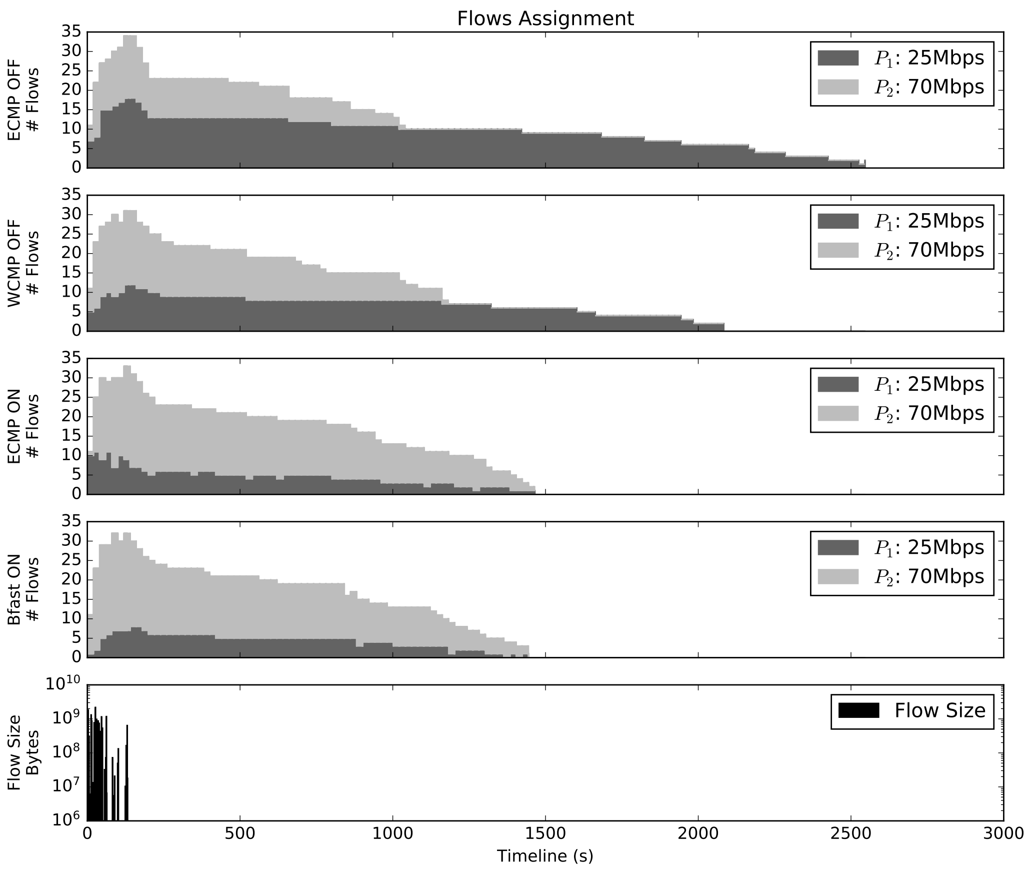


Figure 8: Number of flows for an experiment sending traffic through the Internet for different initial assignment plus rebalance strategies. The bottom subgraph shows flow size and starting time.

6. Experiments with a Bartolomeu discrete-event simulator with content provider traces

In this section we describe a discrete event simulator that is fed with the traces for for all destinations presented in Section 4 to provide an estimation of the gains that can be achieved by the deployment of Bartolomeu in a content provider. Then, we discuss the results. Finally, we present some scalability considerations derived from these results.

6.1. Simulator and experiment description

We simulate a client/server connection with a simple fixed-capacity path model and a discrete-event implementation of the PIM and LBDM modules. LBDM implements ECMP, Bfast and WCMP initial allocation strategies, along with legacy ECMP. We use RNP and WIDE traces to compare and analyze the different flow allocation choices with a real traffic mix in which there are prefixes with different number of flows and flows with different sizes and arrival time. To process the traces, we consider 3 different sampling rates, 1024, 4096 and 16384. We set the observation window W to 20 s for the first two sampling rates and to 40 s for the last one.[4] We use the start time and flow size inferred from the traces as input to the simulation. The path bitrate for a destination prefix p, R_p, is estimated from the traces as the number of bytes transferred for the prefix divided by the time during which at least one flow was active. Each prefix is simulated independently, i.e., as if the bottleneck of the paths for different prefixes were not shared.

We consider the following three network setups:

- Two paths, each one with $R_p/2$.
- Two paths, one with $3R_p/4$, and the other $R_p/4$, so that the first has three times the bitrate of the second. This rate distribution is similar to the one reported in the experiment at Section 5.
- Two paths, one with $10R_p/11$, and the other $R_p/11$, so that the first has ten times the bitrate of the second.

[4]If the optimal observation window W is unknown when configuring the system, we propose assigning 20 s to the observation window W for rates going from 256 to 4096 and 40 s for the rest of the rates. This simple strategy provides good performance for both trace sets.

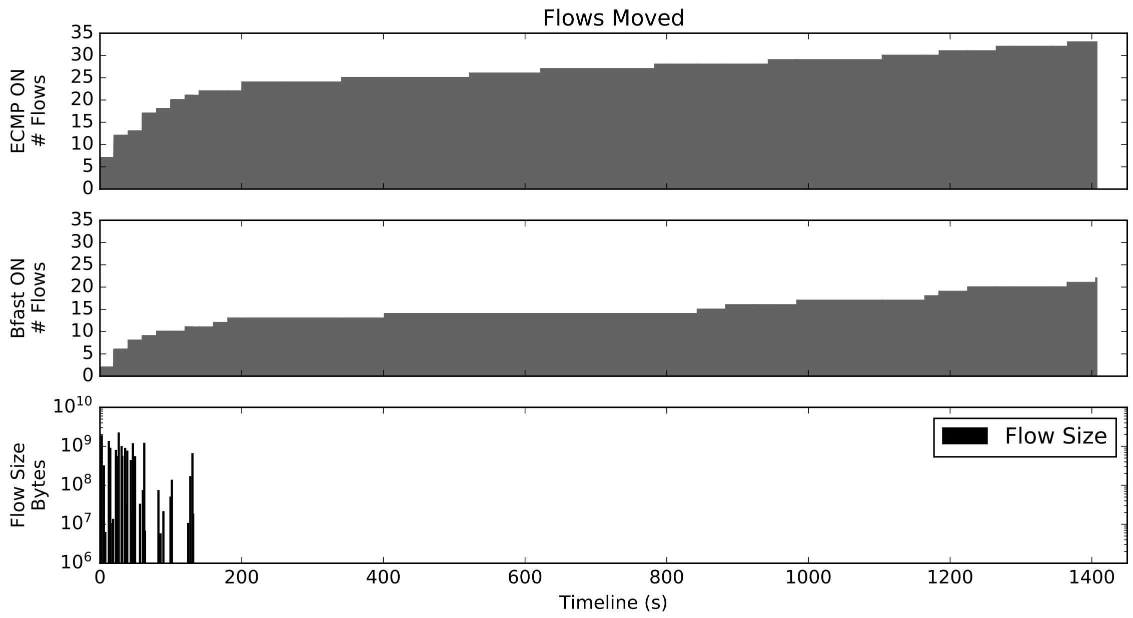


Figure 9: Number of flows moved for the same experiment of Fig. 8, ECMP + REBALANCE and Bfast + REBALANCE cases. The bottom subgraph shows flow size and starting time.

In regard to ECMP, we assume that all the paths are equally eligible, i.e., they traverse the same networks, as this is the condition for Multipath BGP. Note that Bartolomeu can use any available BGP path.

In the simulation, the measured rate for a path is constant over the experiment, and does not depend on the traffic we generate. We set the quantum q that triggers LBDM activation to 20 s. For each configuration (traffic pattern, assignment policy, path rates) we perform 20 rounds in which we vary the random seed to allow create different hashes for WCMP/ECMP.

For each experiment we compute:

- Mean sojourn time for the flows in the experiment.
- Maximum number of flow entries, observed every q seconds, that are installed in the switch to implement the considered policy.
- Number of flow movements over the experiment. This value is higher than the flow entries because entries expire when the flow finishes, and a flow could be moved more than once (thus counting one as an entry, but more than one as flow movements).
- Maximum number of flow entry change requests, observed every q seconds. This represents the number of requests a controller should issue to the switches.

6.2. Results and analysis

Tables 4 and 5 show:

- The ratio between the mean sojourn time for all ECMP experiments (no REBALANCE) and the mean sojourn time for the experiments with different combinations of initial allocation and REBALANCE. Numbers larger than 1 represent a gain of the considered strategy compared to the default non-Bartolomeu configuration.
- Maximum number of flow entries for all the experiments.
- Maximum number of flow movements for all the experiments.
- Maximum for all the experiments of the maximum number of flow entry change requests observed at each one.

We derive the following conclusions from the results:

- REBALANCE is the most influential strategy to reduce the sojourn time for any traffic set and egress rate relation, more important than the strategy for the allocation of new flows. This is consistent with the result of the experiment at Section 5.
- The achievable gain grows with the ratio between the egress path rates. For paths with equal rate, low gains (between 3 and 5%) can be achieved. In this case, ECMP is naturally a good strategy, and Bartolomeu gains just come from compensating uneven flow-to-path assignment and different flow duration. Sojourn time is reduced to half when path rates differ by a factor of three. For a large difference in the path rate (10 times), we can reduce the mean sojourn time to a sixth.

S	W	Traffic %	Prefixes	Flows	Initial Assig. / Rebalance	$R1 = R2$				$R1 = 3 \times R2$				$R1 = 10 \times R2$			
						Ratio	Entries	Moved	Reqs	Ratio	Entries	Moved	Reqs	Ratio	Entries	Moved	Reqs
1024	20	68.69	6529	36842	WCMP off	1.000	0	0	0	1.313	0	0	0	2.650	0	0	0
					ECMP on	1.055	5695	8321	69	2.039	17783	20645	265	5.856	21994	23829	304
					Bfast on	1.050	9642	11326	87	2.082	4576	7395	59	6.120	2583	4982	41
4096	20	57.34	4066	15001	WCMP off	1.000	0	0	0	1.293	0	0	0	2.543	0	0	0
					ECMP on	1.038	2341	3204	32	1.998	9489	10369	141	5.677	10798	11348	144
					Bfast on	1.034	3652	4161	36	2.055	1748	2811	29	6.008	1009	1999	21
16384	40	39.17	2011	4878	WCMP off	1.000	0	0	0	1.275	0	0	0	2.414	0	0	0
					ECMP on	1.028	614	815	12	1.991	3757	3992	46	5.565	4064	4193	46
					Bfast on	1.025	844	1004	13	2.039	437	776	10	5.829	295	626	10

Table 4
Results for Bartolomeu discrete-event simulator with RNP traces.

S	W	Traffic %	Prefixes	Flows	Initial Assig. / Rebalance	$R1 = R2$				$R1 = 3 \times R2$				$R1 = 10 \times R2$			
						Ratio	Entries	Moved	Reqs	Ratio	Entries	Moved	Reqs	Ratio	Entries	Moved	Reqs
1024	20	66.80	2463	28063	WCMP off	1.000	0	0	0	1.313	0	0	0	2.633	0	0	0
					ECMP on	1.058	3379	5781	92	2.038	8589	10757	198	5.805	11638	13011	213
					Bfast on	1.054	7282	8798	142	2.069	3357	5170	72	6.002	1620	3108	47
4096	20	59.00	1623	9681	WCMP off	1.000	0	0	0	1.304	0	0	0	2.601	0	0	0
					ECMP on	1.048	1420	2288	40	2.004	4223	5097	89	5.685	5154	5682	92
					Bfast on	1.045	2466	3072	48	2.046	1257	2090	27	5.926	696	1411	21
16384	40	48.31	815	2686	WCMP off	1.000	0	0	0	1.309	0	0	0	2.608	0	0	0
					ECMP on	1.059	481	782	16	2.034	1691	1987	44	5.749	1895	2084	50
					Bfast on	1.056	668	894	21	2.072	368	661	12	5.986	221	486	13

Table 5
Results for Bartolomeu discrete-event simulator with WIDE traces.

- Different FIM settings, and thus different number of flows eligible for rebalancing, result in very similar mean sojourn ratios. However, we note that the amount of traffic managed, and thus the amount of traffic benefiting from Bartolomeu, varies.

- The mean sojourn time ratio is very similar for the same PIM and LBDM configuration and ratio of rates, regardless the dataset (RNP and WIDE).

- The maximum values of the number of entries depend on the FIM configuration, and it increases for large amounts of managed traffic. To evaluate the requirements imposed to the hardware of the switch, we have to add to this number the rules required to measure the egress rate for the prefixes with relevant flows. Thus, the maximum number of entries that a switch needs to support is for ECMP with REBALANCE and sampling rate of 1024. In this case, we need 6,529 multiplied by the number of egress links (number of prefixes to monitor by the number of paths), plus 21,994 flow entries observed in a q period, for a total of roughly 35,000 entries.

- The number of flows moved across the experiment is in all cases is smaller than the total number of flows. This implies that the average number of movements a flow experiences is below 1. This is aligned with our objective of ensuring that the mechanism does not result in high flow churn.

- Regarding to the number of entry change requests per second, the maximum value is around 15 operations per second (304 in a 20 s period).

6.3. Feasibility analysis of the deployment of Bartolomeu

In this subsection we analyze how the current technology can support the requirements raised by Bartolomeu's operation, based on the results of the simulation. We discuss how Bartolomeu can be configured to fit into different technological constraints.

The result of our experiment for RNP and WIDE traces indicates a maximum number of rules to install of around 35,000. This number grows with the egress link count, roughly adding a maximum of 6,500 prefixes per egress link. The number of different rules supported by the current generation of SDN switches is around 100,000 [31], while first generation ones limited the number of flow entries to around 2,000 [6]. Thus, the value measured in our experiment is reasonable for current switches, but clearly exceeds first generation ones. Besides, the number of entries to install can be controlled with some of the Bartolomeu operation parameters. In particular, the number of entries can by reduced almost in an order of magnitude by changing the sampling rate from 1,024 to 16,384 (Tables 4 and 5).

Another constraint is the rate at which entry changes requested by the SDN controller can be performed at the switches. We observed a maximum rate of 15 requests per second, well below the 230 flow installations per second supported in the first generation of OpenFlow switches [3]. In case the number of changes required by Bartolomeu were too high, reducing the sampling rate also reduces this value, as Tables 4 and 5 show.

We now analyze the requirements imposed to the controller modules that define Bartolomeu's architecture. We consider that a large number of switches could need to be managed in a coordinated fashion. RIM, in charge of gathering BGP routes, would receive as many BGP feeds as BGP neighbors connected to the egress switches.

These BGP neighbors may generate a large number of updates to process. We already referred in Subsection 2.1 to software architectures that claim to process similar loads with response times in the order of seconds. In addition, the architectures mentioned are scalable, as new resources could be added as the number of neighbors or prefixes grow.

The FIM module is a probe collector responsible for detecting elephant flows. As occurs for RIM, this software can be engineered to be modular, interfacing with different switches, and thus scalable. In addition, the sampling rate can be adjusted to reduce the amount of probes, and the observation window controls the rate at which the elephant detection process is triggered.

PIM periodically receives from each switch the amount of traffic for each ⟨BGP prefix, next-hop⟩ pair with at least one active elephant flow. PIM performs these requests through the SDN controller. Our measurements of the previous subsection (prefixes column in Tables 4 and 5) indicate a maximum count of around 6,500 prefixes over the whole experiment period. This is an upper bound of the number of prefixes that should be monitored at a given time. The requirements for configuring this measure in the switch have been already included in the total rule count when discussing the number of entries to accommodate in the switch. The transference of this data to the controller is performed in a single request that retrieves the information of the active monitoring rules. This process is performed every q seconds (20 seconds in our setup), period that can be extended in case this transference is a bottleneck. The PIM module software component can be modularized to cope with any reasonable load.

Lastly, we discuss LBDM operation. LBDM is in charge of deciding which flow to rebalance and configure the switches. The complexity for the processing of each prefix with active flows is in the order of $O(m^2 + F_h log(F_h))$ (see Subsection 2.4), with m the number of egress paths available for the prefix, and F_h the number of relevant flows per prefix and per path. Neither of them is expected to be a large number. Besides, the LBDM algorithm is defined in a per-prefix basis, so the whole prefix set could be partitioned and processed by different LBDM instances if needed.

7. Related Work

In this section we first analyze contributions related to the application of SDN to manage the BGP protocol in an AS (BGP-SDN) and then some proposals about the distribution of traffic across different paths.

Caesar et al. [7] propose *Routing Control Platform* (RCP), an architecture in which the BGP information of a whole AS is aggregated and processed in a central system. Thus, the routing function is completely separated from the data-plane. Such an architecture can improve scalability and reduce configuration complexity. They show that this solution is feasible for large ASes, as it can manage the whole BGP table (around 200,000 prefixes at the time) and timely process the BGP updates received from 100 different external sources.

SDN-IP [28] implements the aforementioned RCP architecture with OpenFlow. In this case, all the internal elements of the AS are migrated to SDN, including the legacy border routers. Rothenberger et al. [38] describe RFCP (RouteFlow Control Platform), another implementation of centralized BGP processing for an AS. In this case the architecture is hybrid, as it allows native BGP routers to coexist with OpenFlow switches and controllers in the same AS. BTSDN [27] also presents a hybrid solution aiming to facilitate the transition from a traditional routing model to a SDN centralized model. In this architecture, the central BGP controller manages OpenFlow proxy switches that in turn interact with the legacy BGP routers at the border of the network. The architecture presented for Bartolomeu in this paper is a pure BGP-SDN one, similar to the SDN-IP proposal [28], as all elements are controlled according to the SDN paradigm. However, Bartolomeu could be adapted to operate in a hybrid architecture composed of SDN and legacy IP equipment, such as RFCP or BTSDN, as long as (a) there is a central system aware of the whole BGP state for the AS that can determine the egress path for particular flows, and (b) it is possible to measure the egress rate in a per-destination and next-hop basis.

In order to make a scalable BGP-SDN system, Duan et al. proposed a system called OFBGP [12]. Execution task units can be deployed in different computing devices to gather routes from different peers, and to execute the BGP route selection process for different destination prefixes. As the routes to a single destination are processed in the same system, it is a centralized BGP-SDN system. OFBGP provides the glue to coordinate both tasks. In addition, OFBGP also deals with the implementation of high availability mechanisms through BGP Non-Stop-Routing (BGP-NSR) technology. As described earlier in Section 2, Bartolomeu's RIM module is responsible for gathering all BGP route information. RIM could be implemented similarly to OFBGP so that it can easily scale horizontally when the number of BGP neighbors grows.

SDNMA [18] explore the manipulation of paths by a BGP-SDN system. The application is composed of two main modules, a BGP-Speaker module and a Traffic Engineering module (TE). The results show that the TE module can act with real time information, such as throughput, delay, and loss to manipulate the allocation of flows. However, the algorithms used to change the flow paths are not detailed, and a simple proof-of-concept test is described.

Edge Fabric [39] is an SDN-BGP architecture deployed by Facebook to improve egress traffic performance. As Bartolomeu, Edge Fabric has a central control point which gathers information of all the BGP routes received from external peers and real-time traffic information measured at the egress links. This information is used to redirect egress traffic, within the same PoP (Point of Presence), in a periodic fashion. The fundamental difference between Edge Fabric and Bartolomeu is that Edge Fabric assumes that the bottleneck occurs at the egress link, and aims to keep its utilization below 95% while Bartolomeu aims to improve end-to-end performance through per-path and per-destination prefix rate measurement, being able to detect bottlenecks appearing more than one hop away to the content provider. Schlinker et at. [39] also report the RTT, sending rate, etc., measured at server nodes, for random flows routed through alternate paths, i.e., non-BGP preferred paths. They indicate that half of the explored paths result in a better median latency than BGP most preferred ones, so they conclude they could detect and use them in future upgrade of Edge Fabric. We note that these paths are naturally used by Bartolomeu, without the need of server-collected performance metrics.

Espresso [42] is another edge SDN-BGP architecture, in this case used by Google PoPs to route egress traffic to clients. A global controller is in charge of receiving all external BGP routes, link utilization and end-to-end statistics (RTT, goodput, retransmissions, etc.) to assign (per-application specific) traffic to egress paths. While the Espresso paper does not describe how these inputs are combined to determine the selected route for each BGP destination prefix, the algorithm used by Bartolomeu is described in our paper making the results reproducible. Content servers play a major role in Espresso operation, as they generate at the transport and application layer the end-to-end metrics the controller uses to perform path selection. Besides, the servers themselves are responsible of enforcing the egress next-hop by labelling the packets they generate according to an MPLS FIB configured by the controller. As Internet-scale routing is performed by the source nodes, the servers, standard BGP external routers can be replaced by more simple devices just in charge of forwarding at wire speed MPLS-tagged traffic. Bartolomeu is designed to operate in a more general scenario in which servers are not engineered neither to provide performance feedback to the network controller nor to enforce routing themselves.

We next discuss proposals that address the problem of distributing traffic inside an ISP or a datacenter. MATE [13] is the seminal paper regarding to traffic engineering across a network. It is a distributed mechanism that dynamically allocates flows from a given source/destination pair to different existing MPLS paths. Flows are assigned to flow aggregates (called bins) as the result of a hash process. MATE nodes actively measure the latency and packet loss in the paths using probe packets. Upon a change of the measured values, source nodes can reallocate bins to different paths to minimize the total sum of the path latencies in the network. TeXCP [24] is similar to MATE, although it differs in minimizing the maximum link utilization. To achieve this, it requires internal nodes to exchange this information across the network. While these proposals adhere to the idea of dynamically reassigning flows to paths, they differ with Bartolomeu in the information used to trigger the path changes. In the fully-controlled scenario of an ISP, routers can be requested to perform active measurements or to exchange information about link utilization. Note that the availability of this information makes this problem fundamentally different to Bartolomeu's, which does not assume any other information but the one it can measure locally (in the egress links of the AS).

Regarding to datacenter traffic management solutions, we discuss Fat-tree, Hedera, Mahout and MicroTE. Fat-tree [1] relies in several greedy heuristics to distribute flows among paths. When a new flow arrives to a switch, Fat-tree assigns it to the path with the least loaded egress link. Then, every few seconds, the switch triggers a rebalancing process for at most three flows to equalize the sending rates across the egress links. However, paths for large long-lived flows are allocated with a different strategy. In this case, a central scheduler aims to assign long-lived flows to non-conflicting paths (i.e., paths for which no other long-lived flow has been previously assigned) if possible. Hedera [2] assigns long-lived flows similarly to Fat-tree, in the sense that it uses a central scheduler to allocate flows avoiding non-conflicting paths when possible. For the set of identified elephant flows, Hedera first computes a target source-destination bandwidth matrix that takes into account the limitations set by the end hosts. Then, it uses the complete knowledge of the capacity of the paths, and the information about the rest of the flows, to perform flow-to-path assignment. This is equivalent to a *multi-commodity flow* problem, and they propose some heuristics to solve it. Besides, Fat-tree and Hedera long-lived flow management depends on the complete knowledge of the number and source/destination of the flows currently in the network and knowledge of its topology that is not available for Bartolomeu. Bartolomeu is designed to operate without cooperation of any of the intermediate routers in the Internet, or the end hosts.

Mahout [10] is a centrally controlled system for balancing load in datacenters. Mahout aims to allocate elephant flows to least congested paths. Each time a new elephant flow is detected, an SDN controller assigns it to the least congested path according to link utilization information periodically pulled from every switch of the network. This flow is not moved anymore. As commented before, Bartolomeu does not have access to link utilisation over the whole path the traffic traverses, so that our solution is fundamentally different to this.

MicroTE [5] takes advantage from the predictability of the traffic matrix in datacenters. A centralized component measures the traffic exchanged during short periods of time (in the order of one second) between top of the rack (ToR) switches, and analyses a temporal series to determine which of these switch pairs are expected to continue communicating with similar rates. For these switches, and the estimated bandwidth (they call these traffic *predictable*), MicroTE proposes two approaches to allocate paths, in both cases with the objective of minimizing the maximum link utilization (this is similar to Hedera). For the non-predictable traffic, switches are configured to distribute flows proportionally to the rate that was not reserved for the predictable traffic, using WCMP for this purpose. Bartolomeu may also use WCMP for initial traffic allocation, although the weights are configured according to the per-prefix egress rate, instead of depending on the utilization information available in MicroTE.

Replex [15] proposes a mechanism that rebalances traffic periodically that can be applied for interdomain traffic. Its feedback loop is based on path latency and it requires collaboration from the other end, as opposed to Bartolomeu, that relies exclusively in locally measured information.

Multihoming Route Control (MRC) has been used by stub networks to equalize the load among the links connecting to different providers. MRC devices rely on a box that inspects TCP traffic, for example, estimating the TCP handshake time, to infer the path latency with top destinations [29]. With this information, they decide how to split traffic among the paths provided by BGP. Bartolomeu addresses the same problem of balancing the traffic of a stub between different paths. In the MRC case, the information used to decide the traffic split is measured locally, as similar to Bartolomeu. The differences are that Bartolomeu does not inspect per-flow traffic exchanges for a predefined set of destinations, but instead it relies on per-BGP destination prefix rate aggregate measures.

Another MRC flavor operates with stub networks that receive addresses from each of its providers, so that the path that a flow follows depends on the addresses the transport connection uses [21]. Externally initiated communications are balanced by the DNS responses, while internally initiated communications could be manipulated by a NAT device deciding the proper path to use and rewriting the source address to match this path. The boxes controlling the DNS or the NAT configuration typically aim to maximize the minimum available bandwidth of the egressing links, and may also monitorize the flows to determine if an end-to-end path is working or not (so that new communications could be initiated through different paths in case the mechanism infers a path failure). Bartolomeu is intended for stub networks with *provider independent addressing*, so that the egressing path does not depend on the selection of addresses from any specific set. Therefore, Bartolomeu does not only control the initial path assigned to a flow, but it can change it dynamically.

Some work deals with fixing the oscillation problems that may arise when multiple MRC devices serving different stub networks operate in parallel [20, 41]. As the controllers of each MRC device are selfish, it may occur that all devices coincide in rerouting traffic to the best available path. The result is a continuous change in the traffic distribution and suboptimal use of the available resources. Some solutions proposed are the introduction of randomness into the control loop [20] or adaptive filters [41]. Bartolomeu may suffer this problem if the controlled networks share paths. Traffic rerouting depends on the q and ϵ parameters, the first controlling the rate at which traffic is moved, and the second the magnitude of the difference required to trigger a change. Further study is required to asses the impact of oscillation in Bartolomeu-controlled networks. Randomness could also be adopted to circumvent this problem.

8. Conclusion

Bartolomeu is a novel technique that aims to improve the performance of egress flows in stub networks. Bartolomeu defines a BGP-SDN architecture that increases the number of paths available and aims to keep all paths busy by assigning and rebalancing traffic in proportion to the measured egress rate for each path and destination prefix. To be scalable, only large flows are moved. Bartolomeu relies on the deployment of standard technology, such as an BGP-SDN controller, SDN switches and passive traffic monitoring such as sFlow. Besides, it does not require changes in networks other than the stub network adopting it.

Our experiments show that Bartolomeu can greatly reduce the mean sojourn time, i.e., the mean time to flow completion, when the difference in the rate provided by different paths to a destination are large (e.g., reduce this time to half when path rate differences are a factor of 3). The experiments presented show that Bartolomeu provides gain for traffic flows over the Internet, accounting for interference traffic and the impact in TCP of path change. We show with real content provider traces that traffic-to-flow distribution also makes the gains feasible. The experiment results also allow us to conclude that current SDN technology supports the requirements raised by Bartolomeu in terms of controller/switch memory and processing capacity.

9. Source Code and Data Sharing

The software components used in this work are open-source and available for download in [40], along with some anonymized data. We describe these components next:

- The queuing theory model functions used in Section 3.
- The implementation of Bartolomeu's PIM and LBDM used to perform the experiments as described in Section 5.
- The discrete-event simulator, as well as the data reports from RNP and WIDE to allow the reproducibility, used in Section 6. The reports were obtained from the raw data containing information on the relevant flows of each BGP prefix. To preserve privacy, information on the origin and destination of flows has been omitted.

References

[1] Al-Fares, M., Loukissas, A., Vahdat, A., 2008. A scalable, commodity data center network architecture, in: ACM SIGCOMM Computer Communication Review, ACM. pp. 63–74. doi:10.1145/1402946.1402967.

[2] Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A., et al., 2010. Hedera: dynamic flow scheduling for data center networks, in: Nsdi, pp. 19–19.

[3] Appelman, M., de Boer, M., 2012. Performance analysis of OpenFlow hardware. University of Amsterdam, Tech. Rep , 2011–2012.

[4] Arfeen, M.A., Pawlikowski, K., McNickle, D., Willig, A., 2013. The role of the Weibull distribution in internet traffic modeling, in: Proceedings of the 2013 25th International Teletraffic Congress (ITC), IEEE. pp. 1–8. doi:10.1109/ITC.2013.6662948.

[5] Benson, T., Anand, A., Akella, A., Zhang, M., 2011. MicroTE: Fine grained traffic engineering for data centers, in: Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies, ACM. pp. 8:1–8:12. doi:10.1145/2079296.2079304.

[6] Broadcom, . Bcm56850 series. https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56850-series. (Accessed on 01/Aug/2019).

[7] Caesar, M., Caldwell, D., Feamster, N., Rexford, J., Shaikh, A., van der Merwe, J., 2005. Design and implementation of a routing control platform, in: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, USENIX Association. pp. 15–28.

[8] Cho, K., Mitsuya, K., Kato, A., 2000. Traffic data repository at the WIDE project, in: Proceedings of the Freenix Track: 2000 USENIX Annual Technical Conference, June 18-23, 2000, San Diego, CA, USA, pp. 263–270.

[9] Cisco, 2019. Load sharing with BGP in single and multihomed environments. http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13762-40.html. (Accessed on 01/Aug/2019).

[10] Curtis, A.R., Kim, W., Yalagandula, P., 2011. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection, in: Infocom, pp. 1629–1637. doi:10.1109/INFCOM.2011.5934956.

[11] Downey, A.B., 2005. Lognormal and Pareto distributions in the internet. Computer Communications 28, 790–801. doi:10.1016/j.comcom.2004.11.001.

[12] Duan, W., Xiao, L., Li, D., Zhou, Y., Liu, R., Ruan, L., Xia, Y., Zhu, M., 2014. OFBGP: a scalable, highly available BGP architecture for SDN, in: 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems, IEEE. pp. 557–562. doi:10.1109/MASS.2014.124.

[13] Elwalid, A., Jin, C., Low, S., Widjaja, I., 2001. MATE: MPLS adaptive traffic engineering, in: Proceedings IEEE INFOCOM 2001, IEEE. pp. 1300–1309 vol.3. doi:10.1109/INFCOM.2001.916625.

[14] Ephremides, A., Varaiya, P., Walrand, J., 1980. A simple dynamic routing problem. IEEE transactions on Automatic Control 25, 690–693. doi:10.1109/TAC.1980.1102445.

[15] Fischer, S., Kammenhuber, N., Feldmann, A., 2006. REPLEX: dynamic traffic engineering based on wardrop routing policies, in: Proceedings of the 2006 ACM CoNEXT conference, ACM. pp. 1:1–1:12. doi:10.1145/1368436.1368438.

[16] Gallagher, M., 1992. Comparing proportional representation electoral systems: Quotas, thresholds, paradoxes and majorities. British Journal of Political Science 22, 469–496. doi:10.1017/S0007123400006499.

[17] Gämperli, A., Kotronis, V., Dimitropoulos, X.A., 2014. Evaluating the effect of centralization on routing convergence on a hybrid bgp-sdn emulation framework, in: Proceedings of the 2014 ACM Conference on SIGCOMM, ACM, New York, NY, USA. pp. 369–370. doi:10.1145/2619239.2631458.

[18] Gandotra, R., Perigo, L., 2018. SDNMA: A Software-Defined, Dynamic Network Manipulation Application to Enhance BGP Functionality, in: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE. pp. 1007–1014.

[19] Gao, R., Blair, D., Dovrolis, C., Morrow, M., Zegura, E., 2007. Interactions of intelligent route control with tcp congestion control, in: International Conference on Research in Networking, Springer. pp. 1014–1025.

[20] Gao, R., Dovrolis, C., Zegura, E.W., 2006. Avoiding oscillations due to intelligent route control systems, in: INFOCOM, IEEE. pp. 1–12. doi:10.1109/INFOCOM.2006.124.

[21] Guo, F., Chen, J., Li, W., Chiueh, T.c., 2004. Experiences in building a multihoming load balancing system, in: IEEE INFOCOM 2004, IEEE. pp. 1241–1251. doi:10.1109/INFCOM.2004.1357010.

[22] Hassan, S., Arlimatti, S., Elbreiki, W., Habbal, A., 2016. Border gateway protocol based path vector mechanism for inter-domain routing in software defined network environment, in: Open Systems (ICOS), 2016 IEEE Conference on, IEEE. pp. 76–80. doi:10.1109/ICOS.2016.7881992.

[23] Juniper, 2019. Understading BGP multipath. https://www.juniper.net/documentation/en_US/junos/topics/concept/bgp-multipath-understanding.html. (Accessed on 01/Aug/2019).

[24] Kandula, S., Katabi, D., Davie, B., Charny, A., 2005. Walking the tightrope: Responsive yet stable traffic engineering, in: ACM SIGCOMM Computer Communication Review, ACM. pp. 253–264. doi:10.1145/1080091.1080122.

[25] Kleinrock, L., 1967. Time-shared systems: a theoretical treatment. J. ACM 14, 242–261. doi:10.1145/321386.321388.

[26] Kotronis, V., Gämperli, A., Dimitropoulos, X.A., 2015. Routing centralization across domains via SDN: A model and emulation framework for BGP evolution. Computer Networks 92, 227–239. doi:10.1016/j.comnet.2015.07.015.

[27] Lin, P., Bi, J., Hu, H., 2016. BTSDN: BGP-Based transition for the existing networks to SDN. Wireless Personal Communications 86, 1829–1843. doi:10.1007/s11277-015-3145-0.

[28] Lin, P., Hart, J., Krishnaswamy, U., Murakami, T., Kobayashi, M., Al-Shabibi, A., Wang, K.C., Bi, J., 2013. Seamless interworking of SDN and IP, in: ACM SIGCOMM computer communication review, ACM. pp. 475–476. doi:10.1145/2534169.2491703.

[29] Liu, Y., Reddy, A.N., 2007. Multihoming route control among a group of multihomed stub networks. Computer Communications 30, 3335–3345. doi:10.1016/j.comcom.2006.12.010.

[30] Massoulié, L., Roberts, J.W., 2000. Bandwidth sharing and admission control for elastic traffic. Telecommunication systems 15, 185–201. doi:10.1023/A:1019138827659.

[31] Mellanox, . Mellanox spectrum™-2 ethernet switch ic. http://www.mellanox.com/related-docs/prod_silicon/PB_Spectrum-2.pdf. (Accessed on 01/Aug/2019).

[32] Mori, T., Uchida, M., Kawahara, R., Pan, J., Goto, S., 2004. Identifying elephant flows through periodically sampled packets, in: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, ACM. pp. 115–120. doi:10.1145/1028788.1028803.

[33] Nabe, M., Murata, M., Miyahara, H., 1998. Analysis and modeling of world wide web traffic for capacity dimensioning of internet access lines. Performance evaluation 34, 249–271. doi:10.1016/S0166-5316(98)00040-6.

[34] Panchen, S., Phaal, P., McKee, N., 2001. InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks.

[35] Rekhter, Y., Hares, S., Li, T., 2006. A Border Gateway Protocol 4 (BGP-4). RFC 4271. doi:10.17487/RFC4271.

[36] RIPE, . Updates to the RIPE NCC Routing Information Service. https://labs.ripe.net/Members/colin_petrie/updates-to-the-ripe-ncc-routing-information-service. (Accessed on 01/Aug/2019).

[37] RNP, 2019. RNP - Rede Nacional de Ensino e Pesquisa. http://www.rnp.br. (Accessed on 01/Aug/2019).

[38] Rothenberg, C.E., Nascimento, M.R., Salvador, M.R., Corrêa, C.N.A., Cunha de Lucena, S., Raszuk, R., 2012. Revisiting routing control platforms with the eyes and muscles of software-defined networking, in: Proceedings of the first workshop on Hot topics in software defined networks, ACM. pp. 13–18. doi:10.1145/2342441.2342445.

[39] Schlinker, B., Kim, H., Cui, T., Katz-Bassett, E., Madhyastha, H.V., Cunha, I., Quinn, J., Hasan, S., Lapukhov, P., Zeng, H., 2017. Engineering egress with edge fabric: Steering oceans of content to the world, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ACM. pp. 418–431. doi:10.1145/3098822.3098853.

[40] Torres-jr, P.R., 2019. Bartolomeu. http://www.bitbucket.org/torresweb/bartolomeu/. (Accessed on 01/Aug/2019).

[41] Yannuzzi, M., Masip-Bruin, X., Marin-Tordera, E., Domingo-Pascual, J., Fonte, A., Monteiro, E., 2008. Improving the performance of route control middleboxes in a competitive environment. IEEE network 22, 56–64. doi:10.1109/MNET.2008.4626233.

[42] Yap, K.K., Motiwala, M., Rahe, J., Padgett, S., Holliman, M., Baldus, G., Hines, M., Kim, T., Narayanan, A., Jain, A., et al., 2017. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ACM. pp. 432–445. doi:10.1145/3098822.3098854.

[43] You, L., Wei, L., Luo, J., Jian, J., Nu, X., 2015. An inter-domain multi-path flow transfer mechanism based on SDN and multi-domain collaboration, in: IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015, pp. 758–761. doi:10.1109/INM.2015.7140369.

[44] Zhou, J., Tewari, M., Zhu, M., Kabbani, A., Poutievski, L., Singh, A., Vahdat, A., 2014. WCMP: weighted cost multipathing for improved fairness in data centers, in: Ninth Eurosys Conference 2014, EuroSys 2014, Amsterdam, The Netherlands, April 13-16, 2014, pp. 5:1–5:14. doi:10.1145/2592798.2592803.