University of Würzburg
Institute of Computer Science
Research Report Series

Accuracy and Dynamics of Multi-Stage Load Balancing for Multipath Internet Routing

Rüdiger Martin, Michael Menth, Michael Hemmkeppler

Report No. 419 April 2007

University of Würzburg
Institute of Computer Science
Department of Distributed Systems
Am Hubland, 97074 Würzburg, Germany
{martin,menth,hemmkeppler}@informatik.uni-wuerzburg.de

Accuracy and Dynamics of Multi-Stage Load Balancing for Multipath Internet Routing

Rüdiger Martin, Michael Menth, Michael Hemmkeppler
University of Würzburg
Institute of Computer Science
Department of Distributed Systems
Am Hubland, 97074 Würzburg, Germany
{martin,menth,hemmkeppler}@informatik.uni-wuerzburg.de

Abstract

Flow-based load balancing algorithms for multipath Internet routing are often used for traffic engineering. However, the target load distribution and the load balanced result agree only on average, and there is a significant inaccuracy over time due to stochastic effects. Dynamic load balancing reduces this inaccuracy by relocating flows to other paths in regular time intervals. This causes packet reordering. Therefore, the flow reassignment rate should be kept low. In this paper we consider load balancing in networks. It differs from load balancing at a single node by the fact that several load balancing steps may be performed at consecutive nodes in series. This affects the flow reassignment rate and the load balancing accuracy due to interdependencies and polarization effects. We quantify the impact by simulation results, explain the observed phenomena, and give recommendations for load balancing in practice.

1 Introduction

Multipath Internet routing is used, e.g., for traffic engineering to distribute the traffic more evenly through the network. This requires load balancing algorithms to spread traffic with the same destination over several interfaces. Load balancing should be done per flow and not per packet to avoid packet reordering and a detrimental impact on the throughput of TCP [1–3]. Therefore, hash-based load balancing algorithms are used, whose basic architecture is presented in [4]. As flows come and go, the traffic distribution result of the load balancer changes and, as a consequence, the outcome deviates from the intended target distribution. To limit the inaccuracy of the load balanced result, dynamic load balancing algorithms correct the result by reassigning flows to other paths. This causes a route change for these flows and a chance for packet reordering. Therefore, the flow reassignment rate of dynamic load balancing algorithms should be kept low.

In [4] we have considered the accuracy and dynamics of hash-based load balancing algorithms at a single node. This reveals the general properties of the algorithms and is relevant if load balancing is applied to traffic aggregates only once. A typical application example is the self-protecting multipath [5] (SPM). The SPM transmits the traffic over several disjoint paths according to a load balancing function (cf. Fig. 1). In case of a path failure, the flows are redistributed from the failed path to the working paths

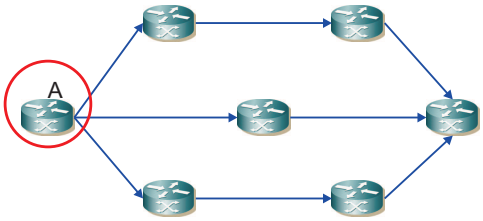

Figure 1: The SPM load balances the traffic only once.

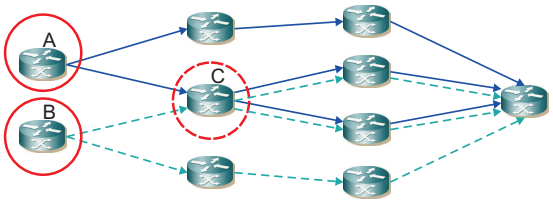

Figure 2: ECMP routing causes the traffic to undergo multiple load balancing steps.

according to another path-failure-specific load balancing function. Load balancing is also required for equal cost multipath (ECMP) routing with OSPF [6], IS-IS [7], or some proprietary RIP implementations [8]. This application scenario differs from the SPM by the fact that traffic undergoes load balancing possibly more than once and that the amount of input traffic for a load balancer depends on preceding load balancers, which is illustrated by router C in Fig. 2. This creates two new problems: (1) flows forwarded by an earlier hash-based load balancer over a specific interface are "polarized" such that a succeeding load balancer is potentially not able to spread this traffic aggregate anew [9]; (2) flow reassignments by a preceding dynamic load balancer entails possibly further flow reassignments at succeeding load balancers since suddenly missing or new flows affect their traffic distribution. In this paper, we study how the load balancing accuracy and the flow reassignment rate is affected by these issues.

The paper is structured as follows. Section 2 gives an overview on static and dynamic hash-based load balancing algorithms. Section 3 explains our simulation model and reviews the problems of single-stage load balancing while Section 4 presents our new results regarding the accuracy and the dynamics of multi-stage load balancing. Finally, we summarize our work in Section 5.

2 Overview of Hash-Based Load Balancing Algorithms

The following notation formalizes the problem of load balancing for multipath routing. The set of outgoing links (interfaces) $\mathcal{L}(r,d)$ at router r to destination d can be derived from the routing table and corresponds to the paths used from r to d. All flows at a certain router r with destination d are denoted by the flow set $\mathcal{F}(r,d)$. The destination d actually represents the set of destinations subsumed by one entry in the routing table. Hence, the flows in $\mathcal{F}(r,d)$ are all spread over the same interfaces. The target load fraction $tLF(r,d,l)$ for a specific outgoing link $l \in \mathcal{L}(r,d)$ describes the desired load balancing objective as a percentage of the total traffic forwarded at router r towards destination d over link l. Thus, the condition $\sum_{l \in \mathcal{L}(r,d)} tLF(r,d,l) = 1$ must be fulfilled. For instance, if router r uses two outgoing links l_0 and l_1 to spread the traffic towards d equally, then $\mathcal{L}(r,d) = \{l_0, l_1\}$ and $tLF(r,d,l_0) = tLF(r,d,l_1) = 50\%$.

Hash-based load balancing algorithms first use a hash function $h(\cdot)$ and a characteristic flow ID $id(f)$ of a flow f to compute a hash value $h(id(f))$. A link selector function $s_{r,d}(h(id(f)))$ then yields the outgoing interface $l \in \mathcal{L}(r,d)$ from the respective set of outgoing links. This functional approach avoids the need to store the corresponding outgoing interface for every flow separately. The authors of [10] analyzed different hash functions for this purpose. We use the 16-bit cyclic redundancy check (CRC) in our experiments as recommended in their study. The flow ID $id(f)$ consists mostly of the five-tuple source and destination IP address, source and destination port number, as well as protocol id, or a subset thereof, which are part of the invariant header field of each packet. Thus, hash-based algorithms differ with respect to the applied hash function h and link selector functions $s_{r,d}$.

We assume that the current traffic rate $cTR(r,d,l)$ at router r over a specific link $l \in \mathcal{L}(r,d)$ to destination d can be obtained by some means, e.g. by online measurements [11]. It allows to calculate the current load fraction $cLF(r,d,l) = \frac{cTR(r,d,l)}{\sum_{l' \in \mathcal{L}(r,d)} cTR(r,d,l')}$. If it differs substantially from the target load fraction $tLF(r,d,l)$ due to stochastic effects, a change of the link selector function $s_{r,d}$ is required. For instance, if currently $cLF(r,d,l_0) = 40\% < 50\% = tLF(r,d,l_0)$ and $cLF(r,d,l_1) = 60\% > 50\% = tLF(r,d,l_1)$ for the example from above, then flows should be relocated from l_1 to l_0 to abolish this imbalance.

2.1 Static and Dynamic Load Balancing Algorithms

Static load balancing algorithms do not allow such a change of the link selector function $s_{r,d}$ while dynamic algorithms automatically adapt their link selector function to achieve a new balanced traffic distribution.

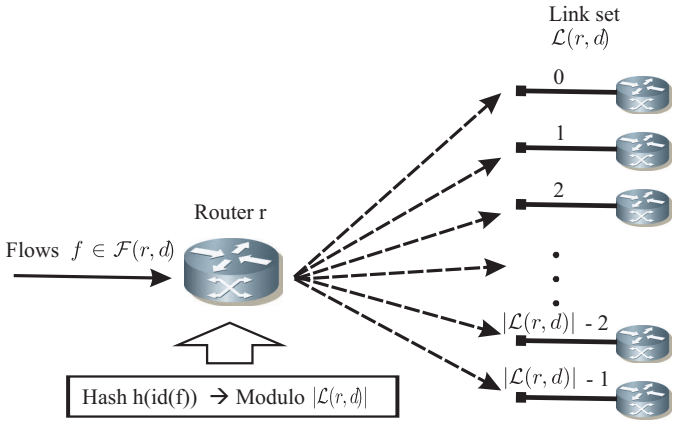

Figure 3: Data structure of a direct link selector function.

2.1.1 Static Hashing

Link selector functions perform either a direct mapping between hash values and links or an indirect, table-based mapping using intermediate data structures.

Direct Hashing Direct link selector functions may be implemented by a simple modulo operation, i.e., $\mod (h(id(f)), |\mathcal{L}(r,d)|)$ determines the number of the outgoing interface within the link set. This leads to an even objective distribution of the traffic aggregate $\mathcal{F}(r,d)$ over the links in $\mathcal{L}(r,d)$: $tLF(r,d,l_i) = tLF(r,d,l_j)\ \forall l_i, l_j \in \mathcal{L}(r,d)$. The data structure of such a direct link selector function is illustrated in Figure 3.

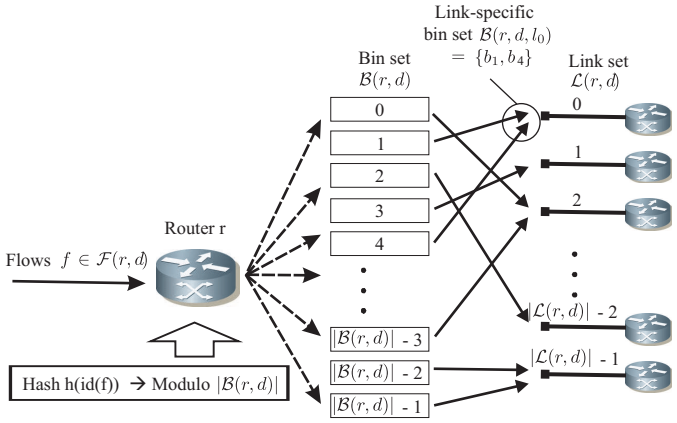

Figure 4: Data structure of a table-based link selector function.

Table-Based Hashing Target load fractions other than even load distribution can be obtained by table-based link selector functions. They perform an indirect mapping from the hash value $h(id(f))$ to an outgoing interface $l \in \mathcal{L}(r,d)$ via so-called intermediate bins. The bins have pointers to the outgoing interfaces. The entire bin set is denoted by $\mathcal{B}(r,d)$ and the bins are numbered 0, ..., $(|\mathcal{B}(r,d)| - 1)$. Now, the table-based link selector function consists of a bin selector function (e.g. $mod(h(id(f)), |\mathcal{B}(r,d)|)$) that maps a hash value to a specific bin, and the pointer of the bin that further directs the flow f to an interface. The data structure of such a table-based link selector function is illustrated in Figure 4. The link specific bin set $\mathcal{B}(r,d,l)$ contains all bins of $\mathcal{B}(r,d)$ with pointers to l.

2.1.2 Dynamic Hashing

For static link selector functions, the assignment between bins and links is fixed. Dynamic algorithms adapt their link selector functions to the current load conditions during runtime. Increasing the link specific bin set $\mathcal{B}(r,d,l)$ of a link l increases also the current load fraction of l. This is achieved by redirecting pointers to l from bins with pointers

to other links. The reduction of the current load fraction of a link l works analogously. Dynamic algorithms check the current load difference

$$cLD(r,d,l) = cLF(r,d,l) - tLF(r,d,l) \tag{1}$$

for any link $l \in \mathcal{L}(r,d)$ from time to time, e.g. in periodic intervals of length $t_r = 1$ s, and reassign the pointers of the bins if needed. Links with a positive $cLD(r,d,l)$ are called overloaded and those with a negative $cLD(r,d,l)$ are called underloaded. In the example from above, link l_0 is underloaded with a current load difference $cLD(r,d,l_0) = cLF(r,d,l_0) - tLF(r,d,l_0) = 40\% - 50\% = -10\%$. Link l_1 is overloaded with $cLD(r,d,l_1) = 10\%$. A link l may be overloaded with regard to some flow set $\mathcal{F}(r,d)$ and, simultaneously, it may be underloaded with regard to some other flow set towards other destinations.

2.2 Hash-Based Load Balancing Algorithms under Study

In [4] we introduced a modular composition of load balancing algorithms based on algorithms from literature and on new ones. The reassignment can be decomposed into a bin disconnection and a bin reconnection step. We proposed various algorithms consisting of a combination of different disconnection and reconnection strategies and evaluated their performance at a single node. Some of the algorithms are simple, others are rather complex – depending on the number of reassigned bins. For the performance analysis of multi-stage load balancing, we use the algorithms with the highest load balancing accuracy from both categories. Both algorithms are greedy. They are only heuristics and achieve certainly not the optimal accuracy. However, simplicity and fast execution counts more than optimality.

In the following, the size of a bin $b \in \mathcal{B}(r,d)$ is determined by its current traffic rate $cTR(r,d,b)$. It is the overall rate of the flows $f \in \mathcal{F}(r,d)$ whose IDs $id(f)$ are mapped to b via the hash and the modulo function. The current traffic load fraction of a bin is defined by $cLF(r,d,b) = \frac{cTR(r,d,b)}{\sum_{b' \in \mathcal{B}(r,d)} cTR(r,d,b')}$. This definition is analogous to the definitions for links.

2.2.1 Single Bin Disconnection (SBD^+)

The single bin disconnection strategy (SBD) is illustrated in Fig. 5. It disconnects from the link with the largest overload the largest bin b that does not turn the link into underload. Then, it reconnects the bin to the link $l' \in \mathcal{L}(r,d)$ with the largest underload. If such a bin b does not exist, nothing is done. SBD^+ avoids to bring any link into underload and is therefore called conservative (+). This avoids heavy oscillations when big bins that turn links into underload are moved back and forth between a few links at successive reassignment steps.

2.2.2 Multiple Bin Disconnection (MBD^-)

The multiple bin disconnection strategy (MBD) is illustrated in Fig. 6. In contrast to SBD^+, the multiple bin disconnection strategy (MBD) disconnects so many bins from

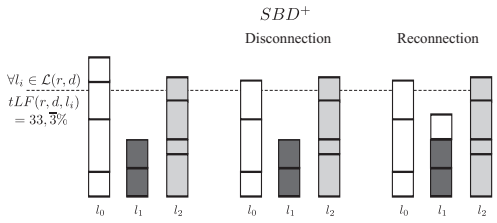

Figure 5: The single bin disconnection SBD^+ relocates only one bin in each step to achieve equal load for the three links l_0,l_1, and l_2.

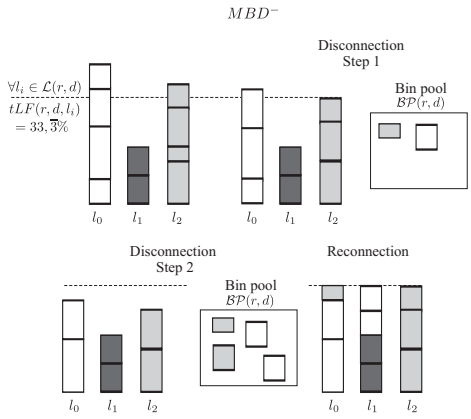

Figure 6: The multiple bin disconnection MBD^- relocates several bins in each step to achieve equal load for the three links l_0,l_1, and l_2.

all overloaded links until any further bin removal turns them into underload. The bins are checked in the order of decreasing size for removal (step 1). Afterwards, each link is turned into underload intentionally by removing its smallest bin from its link specific bin set $\mathcal{B}(r,d,l)$ (step 2). Therefore, we call this strategy progressive (-). The disconnected bins are collected in a so-called bin pool $\mathcal{BP}(r,d)$. Then, these bins are reassigned again in the order of decreasing size. Although MBD^- turns all links into underload, the problem of heavy oscillations is avoided since MBD^- can disconnect several small bins instead of a big one to achieve that goal.

3 Accuracy and Dynamics of Single-Stage Hash-Based Load Balancing

We first explain our simulation model and, then, review the problems of single-stage load balancing with static and dynamic algorithms.

3.1 Simulation Model

The interarrival time of flows on Internet links are exponentially distributed with rate λ_{IAT} [12–14]. Therefore, we apply the Poisson model for flow arrivals in our simulation. The holding times are identically and independently distributed with a mean value of $E[B] = 90$ s. The resulting offered load can be calculated by $a = \lambda_{IAT} \cdot E[B]$ measured by the pseudo unit Erlang (Erl) and reflects the average number of simultaneous flows. We use synthetically generated flow IDs consisting of the four-tuple source and destination IP address and source and destination port.

In the single-stage performance evaluation, we study the load balancing behavior for

a flow set $\mathcal{F}(r,d)$ at router r destined for d and, thus, simulate the traffic distribution to a given number of interfaces at a single node according to a given target load fraction $tLF(r,d,l)$. In the multi-stage analysis, we extend this study to networks and simulate the traffic distribution to a number of paths at multiple interconnected routers according to the respective target load functions.

Standard simulation techniques were applied to obtain confidence intervals and a high simulation credibility. We simulated so long that the 99% confidence intervals deviate at most 1% from the respective mean values. As they are hardly visible, we do not show them in the following figures.

3.2 Impact of Traffic Properties on the Accuracy of Static Load Balancing

Both the flow rate variability and the number of simultaneous flows influence the load balancing accuracy. If all flows have the same size, the task of load balancing reduces to the problem of distributing the active flows over the paths just according to their number and not to their rate. Heterogenous flow rates complicate this task with an increasing variability. In our study we work with flows with heterogeneous rates of 64 kbit/s and 2048 kbit/s and a mean of 256 kbit/s, which yields a relatively high coefficient of variation of 2.29 [15]. In fact, measurements with real Internet traffic found that a few large flows (elephants) produce 50% to 60% of the total traffic while the rest is due to many small flows (mice) [16,17].

We first study the impact of the number of simultaneous flows in a very simplistic scenario. The load of a flow aggregate $\mathcal{F}(r,d)$ is balanced equally between two links by a static load balancer without flow reassignments. We measure the current load fraction $cLF(r,d,l)$ for each link and capture a time-weighted histogram to assess the behavior over time. Figure 7 shows the resulting distribution functions. The x-axis shows the load fraction on one link l with a granularity of 1%, and the y-axis shows the probability that the observed load fractions are smaller than or equal to a value x on this link l at an arbitrary time instant. The results for the second link are symmetric as we consider load balancing over two links here. The load balancing accuracy is high if the curve increases around the target load fraction $tLF(r,d,l) = 50\%$ with a steep slope. The curves correspond to an offered load of $a = 10^{\{2,3,4\}}$ Erl. It is clearly visible that the load balancing accuracy increases with the number of simultaneous flows. An offered load of 10 Erl is definitely too small for load balancing since we observed almost any load fractions between 0% and 100% and, thus, is not shown here. In the following experiments, we consider an offered load of 100 Erl because it is a moderate aggregation degree and, thereby, more challenging for the load balancing accuracy.

3.3 Accuracy Increase through Dynamic Load Balancing

In case of moderate aggregation level, static load balancing is not accurate enough. Dynamic load balancing algorithms are needed. To study their accuracy, we distribute the traffic over four links with target load fractions of 10%, 20%, 30%, and 40% since this is more demanding for the algorithms. The bin reassignment interval length is set

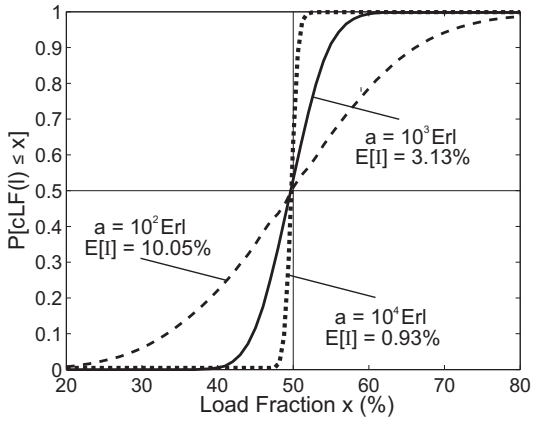

Figure 7: Impact of the offered load on the load balancing accuracy with static hashing (target distribution: $50\%, 50\%$).

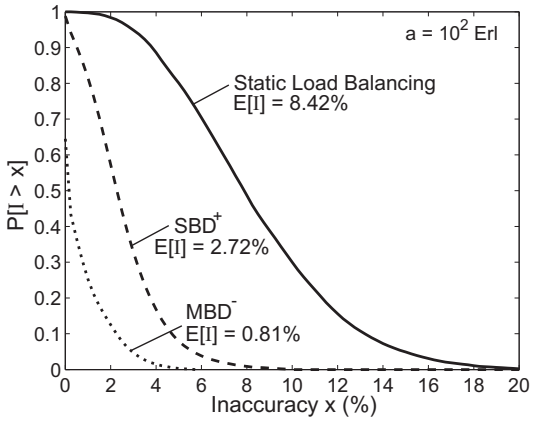

Figure 8: Impact of dynamic algorithms on the load balancing inaccuracy I (target distribution: $10\%, 20\%, 30\%, 40\%$).

to $t_r = 1$ s. We use the average values of the current load difference $cLD(r,d,l)$ (cf. Equation 1) of all links $l \in \mathcal{L}(r,d)$ to measure the load balancing inaccuracy

$$I = \frac{1}{|\mathcal{L}(r,d)|} \sum_{l \in \mathcal{L}(r,d)} |cLD(r,d,l)|. \tag{2}$$

Its mean $E[I]$ captures the inaccuracy over time by a single number. The inaccuracy I is a very intuitive measure, but it only helps compare the algorithms in the same scenario. Load balancing accuracy of scenarios with other target distribution values or even a different number of links cannot be compared by that approach. Figure 8 illustrates the complementary distribution function of I for static hashing, SBD^+, and MBD^-. The faster the curves decay, the higher is the load balancing accuracy. The SBD^+ algorithm ($E[I] = 2.27\%$) is significantly more accurate than static hashing ($E[I] = 8.42\%$) but its accuracy is further improved by the MBD^- algorithm ($E[I] = 0.81\%$). This clearly shows the benefit of dynamic load balancing.

3.4 Drawback of Dynamic Load Balancing

Dynamic load balancing algorithms cause flow reassignments that may lead to packet reordering. Not necessarily every flow reassignment results in packet reordering, but the packet reordering probability scales with the flow reassignment rate $\lambda_{FR}(r,d)$. The flow reassignment rate $\lambda_{FR}(r,d)$ is defined as the average number of reassignments of a flow per second. For a bin reassignment interval length $t_r = 1$ s and MBD^-, the flow reassignment rate is about $0.04\frac{1}{\text{s}}$ which means that a flow is reassigned on average every 25 s and that a flow with a duration of 90 s is reassigned 3.6 times on average. This is

still well acceptable. For SBW^+ it is even lower with a value of about $0.023\frac{1}{\text{s}}$ since only one bin is relocated per reassignment step.

3.5 Impact of Algorithm Parameters on the Accuracy of Load Balancing

The experiments in the preceding paragraph were conducted with 100 intermediate bins. The number of applied bins is a crucial factor for dynamic table-based load balancing algorithms. It directly influences the load balancing granularity. Our performance analysis in [4] showed that a smaller number of bins (10) with dynamic adaptation is counterproductive and large values (500, 1000) do not lead to any further significant improvement. We work with 100 bins because they lead to a sufficiently high accuracy and impose still moderate complexity. Our investigation of the reassignment interval t_r showed that for $t_r \in \{10, 100\}$ s the inaccuracy increases to unacceptably high values and good load balancing results are only achieved for $t_r \in \{0.1, 1\}$ s. However, only for $t_r = 1$ s the flow reassignment rate is acceptable.

3.6 Comparison to other studies

Many related studies (e.g. [18, 19]) perform a fully detailed network simulation on the packet level to measure the packet reordering probability. However, the obtained results depend significantly on the network topology and the routing, on the latency of different paths, and on the queueing delay caused by cross traffic. Thus, there are many other factors but load balancing that influence the packet reordering probability. Therefore, we rather use a flow level simulation and focus on the flow reassignment rate λ_{FR} which is affected only by dynamic load balancing. The packet reordering probability scales with the flow reassignment rate λ_{FR}. Besides, real traffic traces are often used to emphasize that the results are realistic. The quality of hash functions has been examined in [10] with real traffic traces. The 16-bit CRC function that we use in our study spreads the flows most evenly. We study the general potential of different load balancing schemes under various conditions and not the quality of hash functions. Thus, we use synthetically generated flow IDs to avoid any correlation effects within a specific trace.

4 Accuracy and Dynamics of Multi-Stage Load Balancing

We extend the single-stage performance evaluation at a single node to multi-stage in networks where polarization effects and interdependencies between decisions made at different stages occur.

4.1 The Traffic Polarization Effect

With ECMP every node allowing another forking of the multi-path performs load distribution. Thus, traffic undergoes load balancing possibly more than once. This complicates the control over the load balancing result significantly.

In Fig. 9 both router 11 and 21 use the same static load balancing algorithm without flow reassignments. Router 11 ideally splits the flows in half. Since the static load

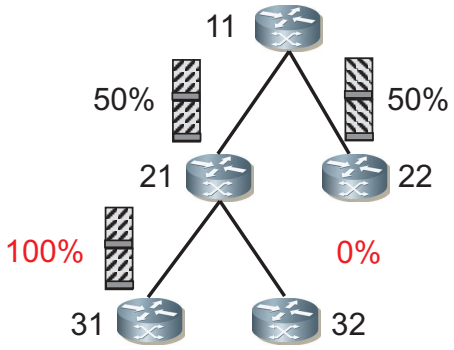

Figure 9: The traffic polarization effect.

balancing depends only on the characteristic flow ID, the algorithms at both routers make the same decisions based on this ID. Every flow that is sent over the left interface by 11 is sent over the left interface by 21 as well since their IDs produce again the same hash values. Thus, the load balancing algorithm at router 21 is without effect. This phenomenon is called polarization effect similar to light passing through polarization filters [9]. Dynamic hashing alleviates this effect as it reassigns flows grouped in bins to other links. However, some bins remain empty and this leads to decreased load balancing granularity and to worse accuracy.

To heal the polarization effect, a randomly generated ID can be assigned to every node in the network. Ideally, this ID is unique for every node and changes the output of the hash function such that the polarization effect vanishes completely. This modification of the input values to the hash function must be fast and retain the original potential of the load balancing mechanisms. We suggest a 32-bit random ID. There are many different possible operations to combine the random ID and the flow ID to a modified input value:

APP Append random and flow ID

XOR Combine last 32 bits of random and flow ID by bitwise-XOR

AND Combine last 32 bits of random and flow ID by bitwise-AND

ADD Perform integer addition between both IDs as binary numbers

So far anti-polarization mechanisms are proprietary and no information about influencing the hash function input values with the random ID are publicly available. In [9] Cisco suggests the use of algorithmically generated ID which is not further specified.

4.2 Accuracy of Hash-Based Multi-Stage Load Balancing

We use the simple test scenario illustrated in Fig. 10 to efficiently test the effect of the proposed modifications against polarization and to evaluate the accuracy of hash-based multi-stage load balancing. To assess the effectiveness of the modifications against

polarization, we use it as a worst case scenario. All routers perform static hashing since it is most sensitive to traffic polarization. All routers at the lower stages obtain input from one link only with traffic that is possibly polarized. Finally, the link selector function simply decides to map even hash values to one link and odd hash values to the other link. Thus, there are no mechanisms to compensate for polarization.

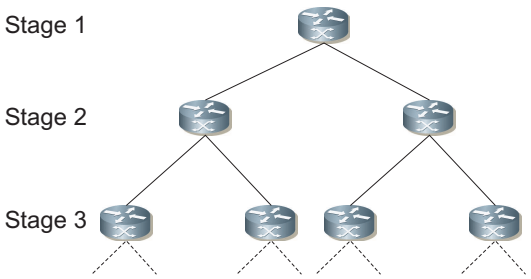

Figure 10: Simple test scenario for the polarization effect.

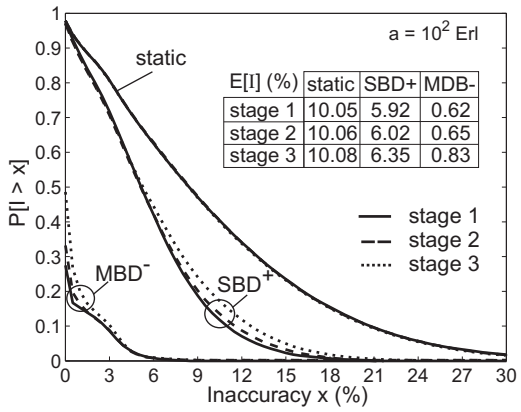

Figure 11: Accuracy of hash-based load balancing algorithms with anti-polarization mechanisms in networks (target distribution: $50\%, 50\%$).

Ideally, the load is split in half at every router. As seen in Section 3.2, the offered load has a severe impact on the load balancing accuracy. For a fair comparison we require an offered load of $a = 10^2$ Erl at all stages where we observe the load balancing results. We achieve this by simulations where we feed the router at the first stage with 100, 200, or 400 Erl when we evaluate the load balancing accuracy on the first, second, or third stage.

Figure 11 shows the complementary distribution function of the load balancing inaccuracy for the bitwise AND and the integer addition on the three different stages together with the mean inaccuracy $E[I] = 10\%$. We omit the results for appending the random ID (APP) and the XOR-operator as they have no effect against polarization. With both APP and XOR one link carries 100% of the traffic at stages 2 and 3. This can be explained by the mathematical properties of the used hash function CRC16. Basically, CRC16 interprets the flow ID as a polynomial over the field consisting of $\{0, 1\}$. The hash value is the residual of the polynomial division of the flow ID by a standardized generator polynomial. Thus, the hash is an element of the vector space of all polynomials of degree at most 16 over $\{0, 1\}$. It can be shown that both modifications are linear functions in this vector space and therefore have no effect on polarization.

The bitwise AND-operator and the integer addition, in contrast, cancel the polarization

effect completely and retain the full load balancing potential of static hashing with $E[I] = 10\%$ at all stages as seen in Fig. 11. These modifications can be interpreted as non-linear functions. Bitwise operations should be preferred as they can be easily computed in hardware. Thus, we choose the bitwise-AND operation to eliminate the polarization effect and use the modified input values in the following experiments if not mentioned otherwise.

Figure 11 also shows the inaccuracy at each stage if we use the dynamic algorithms SBD^+ and MBD^- instead. The load balancing inaccuracy for both algorithms increases slightly at each stage. Thus, even though the polarization vanishes completely as shown above, the dynamic algorithms suffer slightly from the reassignments made at other routers to which they can react after some delay only. However, the loss in accuracy is well acceptable.

4.3 Dynamics of Hash-Based Multi-Stage Load Balancing

To evaluate the dynamics of multi-stage load balancing in terms of flow reassignments, we use the more complex scenario shown in Fig. 12. Flows arrive at the lower stages from two mutually disjoint paths. This models the dynamics caused by multiple independent load balancing entities as nodes in real networks receive traffic from multiple interfaces. At the same time, the symmetry of the scenario still keeps the complexity sufficiently low and we can observe the multi-stage dynamics without bothering with undesirable side effects. Besides, we configure the target load fraction $tLF(r, d, l) = 50\%$ for all routers r and their links $l \in \mathcal{L}(r, d)$. Hence, the routers are expected to receive an offered load of $a = 10^2$ Erl at all stages which does not require different simulation runs for the assessment of the load balancing accuracy at each stage as before. The flow reassignment rates $\lambda_{FR}(r, d)$ are measured locally for each router r. If – for instance – router 11 relocates a flow from the interface to node 21 to the interface to node 22, router 21 perceives this as the termination of the flow. If router 11 changes this assignment later and reroutes the flow to node 21, router 21 perceives this as the start of a new flow.

Figure 13 summarizes the results. The inaccuracy rises slightly from stage to stage for both dynamic algorithms. The gap between stage 1 and 2 is larger than in the previous experiment. This is due to the increased dynamics caused by the input traffic from two independent dynamic load balancing entities. The reassignment rates for SBD^+ remain constant at $0.032\frac{1}{\text{s}}$ because the SBD^+ bin reassignment potential is limited since only one bin is relocated in each reassignment step. For MBD^- the rates increase slightly from stage 1 ($0.031\frac{1}{\text{s}}$) to stage 3 ($0.042\frac{1}{\text{s}}$) due to its larger potential to reassign bins. The increase is still well acceptable. However, for both concepts the overall end-to-end reassignment rate λ_{FR}^{e2e} for the flows routed over the three stages is the sum of the rates at the three stages. Thus, the end-to-end reassignment rate λ_{FR}^{e2e} increases linearly with the number of load balancing stages. Therefore, performing load balancing at too many stages is not recommended.

In addition to the results shown in Fig. 13, we investigated the accuracy and dynamics of SBD^+ and MBD^- in the scenario of Fig. 12 without anti-polarization mechanisms. The polarization effect leads to larger variations among the four different routers at the

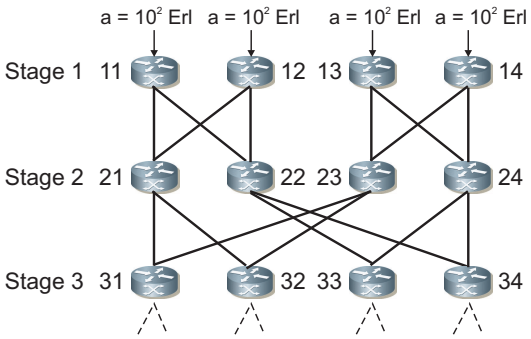

Figure 12: Complex test scenario for the polarization effect.

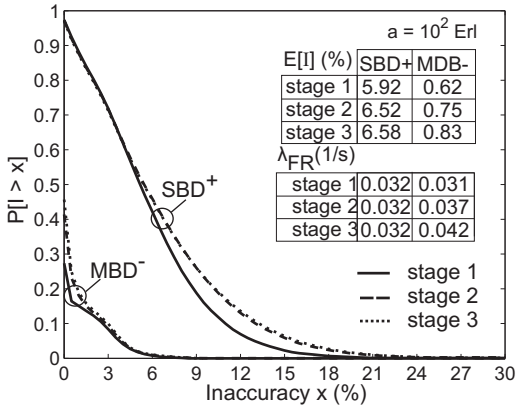

Figure 13: Dynamics of hash-based load balancing algorithms with anti-polarization mechanism in networks (target distribution: $50\%, 50\%$).

same stage than with anti-polarization mechanisms. For instance, the inaccuracy at stage 3 is in the range from $E[I] = 0.72\%$ to $E[I] = 0.94\%$ for the four different routers and the flow reassignment rate in the range from $\lambda_{FR} = 0.043\frac{1}{\text{s}}$ to $\lambda_{FR} = 0.050\frac{1}{\text{s}}$. Thus, polarization leads to performance degradation also in case of dynamic algorithms and the modifications against polarization should be used.

5 Summary and Conclusion

Multipath Internet routing requires load balancing on the flow level to avoid packet reordering. This can be done by hash-based load balancing algorithms. We reviewed the basic architecture of such algorithms and, in particular, explained a simple and a complex load balancer that we identified as especially well performing at single nodes in [4]. They were the candidates for our study. We showed that there is a difference between the target load distribution and the load balanced result due to stochastic effects. Dynamic load balancing mechanisms reduce the inaccuracy by reassigning flows to other paths and cause thereby another potential for packet reordering. We identified traffic properties that influence their accuracy and proposed appropriate parameters for the load balancing algorithms to control it.

In this paper we considered load balancing in networks, i.e. the impact of several load balancing steps in series on the load balancing accuracy and the flow reassignment rate. We explained why simple application of the same load balancing algorithm in case of static load balancers cannot balance the traffic and why this increases the load balancing inaccuracy for dynamic load balancers. We selected an efficient anti-polarization

mechanism among some intuitive candidates and showed that suitable methods provide a general improvement of load balancing methods for their application in networks in terms of accuracy. Then, we investigated the flow reassignment rate in a complex multi-stage network architecture where load balanced traffic from different origins provides the input for the next load balancer. This does not degrade the load balancing accuracy if anti-polarization mechanisms are used, but the overall flow reassignment rate increases approximately linearly with the number of load balancing steps.

After all, load balancing mechanisms should be carefully chosen to minimize the load balancing inaccuracy. Their inaccuracy should be taken into account by the network's resource management, especially if the traffic load is low or moderate. If flows undergo load balancing several times during transportation, anti-polarization mechanisms should be used to get an effective traffic distribution. Finally, load balancing should not be applied too often to the same set of flows since this increases the probability for route flaps and packet reordering.

Acknowledgment

The authors would like to thank Prof. Tran-Gia for the stimulating environment which was a prerequisite for that work.

References

[1] V. Paxson, "End-to-End Internet Packet Dynamics," in *ACM SIGCOMM '97*, Cannes, France, September 1997.

[2] M. Laor and L. Gendel, "Effect of packet reordering in a backbone link on applications throughput," *IEEE Network*, vol. 16, no. 5, 2002.

[3] E. Blanton and M. Allman, "On Making TCP More Robust to Packet Reordering," in *ACM Computer Communication Review*, vol. 32, no. 1, Jan 2002, pp. 20–30.

[4] R. Martin, M. Menth, and M. Hemmkeppler, "Accuracy and Dynamics of Hash-Based Load Balancing Algorithms," in *International Conference on Broadband Communication, Networks, and Systems (BROADNETS)*, San José, CA, USA, Oct. 2006.

[5] M. Menth, A. Reifert, and J. Milbrandt, "Self-Protecting Multipaths - A Simple and Resource-Efficient Protection Switching Mechanism for MPLS Networks," in 3^{rd}*IFIP-TC6 Networking Conference (Networking)*, Athens, Greece, May 2004, pp. 526 – 537.

[6] J. Moy, "RFC2328: OSPF Version 2," ftp://ftp.isi.edu/in-notes/rfc2328.txt, April 1998.

[7] *Information technology – Telecommunications and information exchange between systems – Intermediate System to Intermediate System intra-domain routing information exchange for use in conjunction with the protocol for providing the*

connectionless-mode network service(ISO 8473), International Organization for Standardization, ISO/IEC 10589:2002 (also republished as RFC 1142).

[8] D. Thaler and C. Hopps, "RFC2991: Multipath Issues in Unicast and Multicast Next-Hop Selection," http://www.ietf.org/rfc/rfc2991.txt, November 2000.

[9] "Cisco Systems: Cisco Express Forwarding Overview," http://www.cisco.com/warp/public/732/Tech/switching/docs/ cef_ov_final.pdf, May 2004.

[10] Z. Cao, Z. Wang, and E. Zegura, "Performance of Hashing-Based Schemes for Interet Load Balancing," in *Proceedings of IEEE Infocom*, Tel-Aviv, Israel, March 2000.

[11] R. Martin and M. Menth, "Improving the Timeliness of Rate Measurements," in 12^{th} *together with* 3^{rd} *Polish-German Teletraffic Symposium (PGTS)*, Dresden, Germany, Sept. 2004, pp. 145 – 154.

[12] V. Paxson and S. Floyd, "Wide-Are Traffic: The Failure of Poisson Modelling," *IEEE/ACM Transactions on Networking*, June 1995.

[13] J. W. Roberts, "Traffic Theory and the Internet," *IEEE Communications Magazine*, vol. 1, pp. 94–99, January 2001.

[14] T. Dinh, B. Sonkoly, and S. Molnár, "Fractal Analysis and Modeling of VoIP Traffic," in *Proceedings of Networks 2004*, Vienna, Austria, June 2004, pp. 123 – 130.

[15] M. Menth, "Efficient Admission Control and Routing in Resilient Communication Networks," PhD thesis, University of Würzburg, Faculty of Computer Science, Am Hubland, July 2004.

[16] S. Sarvotham, R. Riedi, and R. Baraniuk, "Connection-level analysis and modeling of network traffic," in *ACM SIGCOMM Internet Measurement Workshop*, San Fransisco, CA, USA, November 2001, pp. 99–103.

[17] N. Brownlee and K. Claffy, "Unterstanding Internet traffic streams: Dragonflies and tortoises," *IEEE Communications*, vol. 40, no. 10, pp. 110–117, October 2002.

[18] J.-Y. Jo, Y. Kim, H. J. Chao, and F. Merat, "Internet Traffic Load Balancing using Dynamic Hashing with Flow Volume," in *SPIE ITCom 2002*, Boston, MA, August 2002.

[19] T. W. Chim, K. L. Yeung, and K.-S. Lui, "Traffic distribution over equal-cost-multi-paths," *Computer Networks*, vol. 49, no. 4, pp. 465–475, Nov. 2005.