©2006 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The definitive version of this paper has been published in IEEE International Conference on Broadband Communication, Networks, and Systems (BROADNETS) 2006 10.1109\/broadnets.2006.4374377.

Accuracy and Dynamics of Hash-Based Load Balancing Algorithms for Multipath Internet Routing

Ruediger Martin, Michael Menth, and Michael Hemmkeppler
Department of Distributed Systems
Institute of Computer Science, University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
Telephone: (+49)-931-8886651, Fax: (+49)-931-8886632
Email: {martin|menth|hemmkeppler}@informatik.uni-wuerzburg.de

***Abstract*— This paper studies load balancing for multipath Internet routing. We focus on hash-based load balancing algorithms that work on the flow level to avoid packet reordering which is detrimental for the throughput of transport layer protocols like TCP. We propose a classification of hash-based load balancing algorithms, review existing ones and suggest new ones. Dynamic algorithms can actively react to load imbalances which causes route changes for some flows and thereby again packet reordering. Therefore, we investigate the load balancing accuracy and flow reassignment rate of load balancing algorithms. Our exhaustive simulation experiments show that these performance measures depend significantly on the traffic properties and on the algorithms themselves. As a consequence, our results should be taken into account for the application of load balancing in practice.**

I. Introduction

Multipath routing is often applied to make data forwarding more robust against network failures [1] and to minimize shared backup capacities [2] when resiliency against network failures is required. It is typically implemented by equal-cost multipath (ECMP) routing in IP networks, e.g., by OSPF [3] or IS-IS [4]. Some proprietary router implementations also offer ECMP-capable RIP [5]. Another option besides native IP is the data transport over disjoint label switched paths (LSPs) using multi-protocol label switching (MPLS) technology. In both cases, load balancing algorithms distribute the traffic over multiple paths towards its destination. In case of LSPs, the traffic is distributed only once over multiple path at the LSP ingress router while with ECMP, every node allowing another forking of the multipath performs load distribution.

Load distribution can be done in a simple way on the packet level, but due to varying link and buffer delays on different paths, this may lead to packet reordering. Since packet reordering severely degrades the throughput of transport layer protocols such as TCP [6]–[8], this is not an option for TCP/IP networks. To avoid packet reordering, all packets of a flow should follow the same path which requires load distribution on the flow level. An intuitive approach maps each individual flow to a certain interface. However, this seriously impedes the forwarding process because the corresponding outgoing interface must be looked up in a table for each packet – let aside the required memory due to the state space explosion. Another approach uses a hash value based on header information to determine the outgoing interface. This makes the storage and the costly lookup of flow information redundant. Hash-based load balancing algorithms can be implemented much easier, but their load balancing accuracy is difficult to control.

The authors of [9] evaluate different hash functions for load balancing algorithms and propose the 16-bit cyclic redundancy check (CRC) function. Several other papers [10]–[13] investigated hash-based load balancing for different applications. Most of them showed that their accuracy is relatively exact for long term averages or they use other performance measures like queue length that are not suitable for our application. Load balancing applied for the optimization of the resource management like in [2] requires a good accuracy of the load balancing at any time instant. To the best of our knowledge, this has not yet been investigated in literature.

In our work, we present well known static and dynamic load balancing algorithms from the literature. Static load balancing algorithms cannot react to load imbalances while dynamic algorithms allow for an adaptive reassociation between hash values and outgoing interfaces. This entails flow reassignments to other interfaces and possibly leads to packet reordering. Thus, the flow reassignment rate should be kept low while the load balancing accuracy is maximized with simple and well implementable algorithms. We suggest a classification of new mechanisms which eventually show better performance than existing ones. Based on this, we conduct a structured analysis of the general load balancing potential. We investigate the impact of dynamic algorithms on the flow reassignment rate and the load balancing accuracy, which allows us to give recommendations for a good algorithm design.

The paper is structured as follows. Section II reviews related work and Section III presents existing static and dynamic load balancing algorithms and suggests a classification of new mechanisms. The simulative performance evaluation of the load balancing accuracy and the flow reassignment rate is done in Section IV. Finally, Section V summarizes this work, draws

conclusions, and gives an outlook on further work.

II. Related Work

Load balancing is used for various application scenarios in communication systems. In general, it distributes service requests to equivalent service entities. First, we illustrate load balancing for multipath Internet routing with its applications and difficulties. Then, we distinguish it from other uses by presenting load balancing for different application examples.

A. Load Balancing for Multipath Internet Routing

Multipath Internt routing occurs whenever packets can be sent over alternative paths. It can be implemented by different algorithms that exhibit algorithm-dependent difficulties.

1) Multipath Internet Routing Applications: Various technical solutions incorporate load balancing.

a) Equal-Cost Multipath Routing: Multipath routing is useful for traffic engineering purposes. In IP networks, it is implemented by the equal-cost multipath (ECMP) routing option. Packets at a certain location are forwarded to their destination over any path with a shortest distance according to the link costs in the network. Multiple paths towards a destination can be obtained by the choice of suitable link costs. ECMP is a standard option of the OSPF [3] and the IS-IS [4] routing protocols. Some proprietary router implementations also allow ECMP with RIP and other routing protocols [5]. Usually, traffic is forwarded equally over any interface leading to the destination over a shortest path. In contrast, adaptive multipath routing [14] is based on relaxed ECMP multipath forwarding structures and signals dynamically the load distribution functions.

b) Resilient Multipath Routing: Resilient multipath routing offers alternative paths such that there is still a working path in case of a failure. This property of multipath routing is deliberately exploited in [1] which is different from the standard IP routing. As long as at least two forwarding alternatives exist, the traffic is distributed in each node according to a given load balancing function.

c) Self-Protecting Multi-Paths: The self-protecting multipath (SPM) consists of disjoint label switched paths (LSPs) and provides at the source several alternatives to forward the traffic to the destination. If one of the paths fails, the traffic is transmitted over the working paths. The traffic distribution over the disjoint path follows an optimized load balancing function to minimize the required backup capacity.

2) Problems due to Load Balancing for Multipath Internet Routing: New problems arise due to the use of load balancing itself or due to the inaccuracy of load balancing.

a) Problems due to the Use of Load Balancing: When multiple paths are used, multiple maximum transfer units (MTUs) may occur on the paths between a pair of nodes [5]. Furthermore, popular debugging utilities like ping and traceroute may become unreliable. Succeeding probes may follow different paths or the diagnosed path does not coincide with the data path. However, the main problem is that different queuing, transmission, and propagation latencies along different paths may lead to packet reordering. Reordered packets have a detrimental effect on the throughput of transport layer protocols like TCP [6]–[8]. Therefore, all packets of a single flow should be forwarded along the same path to avoid packet reordering. Section IV studies the flow reassignment rate that causes packet reordering when load balancing is used.

b) Problems due to Load Balancing Inaccuracy: The resource management entity of a network can configure load balancing to optimize network operation [15]. Then, overload may occur on some links if the achieved load balancing proportions in the network deviate significantly from the corresponding configured values. This is especially problematic if the QoS of real-time traffic is protected by admission control but an unexpected traffic distribution corrupts the planned traffic load in the network [16]. Similarly, backup capacities may not suffice for the SPM or the above mentioned resilient multipath routing if the real traffic distribution in the network deviates from the pre-configured values due to the inaccuracy of load balancing. Section IV investigates this inaccuracy.

3) Load Balancing Concepts: There are various load balancing concepts that can be differentiated regarding their implementation complexity. We give an overview of the principle approaches.

a) Packet-Based Load Balancing: Load balancing can be done through a packet-by-packet assignment of the traffic to the alternative interfaces using an arbitrary scheduling algorithm, e.g. round robin. This packet-based solution is a standard implementation in many state-of-the-art routers but it achieves load distribution on the packet level which causes packet reordering and may entail low TCP throughput.

b) Load Balancing Based on Lookup Tables with Per-Flow-State: An intuitive algorithm to avoid packet reordering is recording an identifier (ID) of a flow together with its outgoing interface in a lookup table. If the first packet of a flow arrives, an interface is selected, the information is inserted into the lookup table, which allows to forward succeeding packets to the same interface. The memory requirements of the table is very expensive for a large number of flows and the lookup in a large table is time-consuming. Therefore, CISCO introduced a limited-size cache [17] and calls it “fast switching”. Whenever the cache is full at the arrival of a new flow, the oldest flow entry of the lookup table is replaced. This possibly leads to packet reordering if this flow is still active.

c) Hash-Based Load Balancing: The problem of large lookup tables can be avoided by hash-based algorithms. A hash function with good statistical properties maps the large space of flow IDs to a smaller space of, e.g., integral numbers. Another operation associates the hash value to outgoing interfaces. No per-flow states are kept since the outgoing interface is derived from the flow ID by mathematical functions. Therefore, hash-based load balancing scales well with an increasing number of flows.

Different hash functions are analyzed in [9]. The authors conclude that the 16-bit cyclic redundancy check (CRC) function [18]–[20] achieves good load balancing performance

among the examined functions for static hashing.

d) Dynamic Load Balancing: The above presented load balancing algorithms were all static in the sense that the mapping between flows and their interfaces is never changed. This makes it hard or even impossible to react to load imbalances. Dynamic load balancing, i.e. flow reassignment to other interfaces, helps to redistribute the traffic load. The authors of [12] developed a dynamic hash-based algorithm that periodically reassigns flows from the most overloaded link to the most underloaded link.

4) Our Contribution: Even though an extensive survey of the load balancing qualities of different hash functions has been presented in [9], there is only little literature about dynamic load balancing for multipath Internet routing. The load balancing accuracy in [12] was estimated based on long-term traffic distributions which lead to the conclusion that the load balancing accuracy is fairly good. This is an intuitive result provided that the hash functions spread large sets of flow IDs evenly over their codomain. Studies of the load balancing accuracy distribution over time are still missing. However, they are required to decide whether forwarding inaccuracies due to load balancing must be considered in the resource management of a network.

B. Load Balancing for Inverse Multiplexing

A single point-to-point link on the network layer may be provided by bundling multiple parallel links on the link layer. The packets of a traffic aggregate are distributed over these parallel links for transmission. This approach is called inverse multiplexing [21] because multiplexing normally means putting multiple small flows onto a large trunk. Typical implementations use packet- or byte-based round robin scheduling [22], which achieves a well balanced load on the separate links.

Due to varying conditions on the single links, packet reordering is also possible like in Section II-A. However, the delay variations are significantly smaller in contrast to load balancing on the IP or MPLS layer. An intelligent packet scheduling at the source allows for efficient packet resequenceing at the sink for point-to-point links. For example, the strIPe protocol does this scheduling and resequencing based on surplus round robin (SRR) on both sides of the physical link [22]. Since multipaths in IP networks may be significantly more complex than parallel links, this solution cannot be adopted for the general problem in Section II-A.

Another implementation approach for inverse multiplexing avoids packet reordering within flows by a hash-based mapping between flows and physical links [13]. The scheme monitors buffer occupancies and reacts to unbalanced load by moving flows from overloaded to underloaded links to prevent packet loss. In contrast, the objective of load balancing for multipath routing is a load distribution according to preplanned values.

C. Load Balancing for Parallel Network Processors on Highspeed Links

Today, highspeed links have such a large bandwidth that network processors are not able to serve them. Therefore, parallel network processors are used to operate a highspeed link at full capacity. The traffic is distributed to different processing units and all packets of a flow should be forwarded to the same network processor to avoid packet reordering. Underloaded network processors lead to underutilized bandwidth and overloaded network processors lead to packet drops.

Like above, hash functions are suggested to map flows with the same hash value to so called flow bundles assigned directly to the processors through lookup table entries [10], [11]. Unbalanced load is detected by monitoring the queue lengths of the network processors and flow bundles are reassigned accordingly. If the time since the last packet arrival is longer than a specified timeout value, the flow bundle may be reassigned to another network processor. Setting the timeout value larger than the packet forwarding latency through the network processor avoids packet reordering. This idea is not applicable to load balancing for multipaths on the IP or MPLS layer because the path latencies can be substantially longer than the one of a network processor.

D. Load Balancing for WWW Caches

WWW caches are used in networks to reduce the number of outgoing WWW requests and to reduce the response time perceived by the users. When caches are distributed over several machines, a hash function maps the request string efficiently to the cache which is responsible for the request. The main focus of this kind of load balancing is not an even load distribution but the reduction of search time and hit rate increase. In case of a cache failure, the authors of [23] developed an elegant algorithm called "highest random weight (HRW)" to deviate requests from the failed cache to other caches. Here, a random weight is calculated for each cache by a hash function based on the request string and the cache ID. The cache with the highest random weight is responsible for the request. If a cache fails, the request points automatically to the cache with the next highest weight.

III. An Overview of Hash-Based Load Balancing Algorithms

The problem of load balancing for multipath routing can be described by the following notation. All flows at a certain router r with destination d are denoted by the flow set $\mathcal{F}(r,d)$. Due to multipath forwarding, there is a set of outgoing links (interfaces) $\mathcal{L}(r,d)$ over which the traffic may be sent. It can be derived from the forwarding table. The desired load balancing for the flow set $\mathcal{F}(r,d)$ is described by the target load fraction $tLF(r,d,l)$, which indicates the percentage of the traffic rate that should be forwarded over $l \in \mathcal{L}(r,d)$. Thus, $\sum_{l\in\mathcal{L}(r,d)} tLF(r,d,l)=1$ is fulfilled.

The ID $id(f)$ of a flow f consists mostly of the five-tuple source and destination 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. Hash-based load balancing algorithms use a hash function $h(id(f))$ to calculate a hash value based on the flow ID $id(f)$. We use the 16-bit CRC as a hash function in our experiments. A

link selector function $s_{r,d}(h(id(f)))$ yields the corresponding outgoing interface $l \in \mathcal{L}(r,d)$ from the respective set of outgoing links for a flow $f \in \mathcal{F}(r,d)$. Thus, hash-based algorithms differ with respect to the applied hash and link selector functions h and $s_{r,d}$.

We assume that the current traffic rate $cTR(r,d,l)$ over a specific link $l \in \mathcal{L}(r,d)$ of a flow set $\mathcal{F}(r,d)$ can be obtained by some means, e.g. by online measurements [24]. The current load fraction is then computed by $cLF(r,d,l) = \frac{cTR(r,d,l)}{\sum_{l' \in \mathcal{L}(r,d)} cTR(r,d,l')}$. If it differs due to stochastic effects substantially from the target load fraction $tLF(r,d,l)$, a change of the link selector function $s_{r,d}$ is required. Static hash-based load balancing algorithms do not allow such a change while dynamic hash-based algorithms automatically adapt their link selector function $s_{r,d}$ to achieve a balanced traffic distribution.

In this section, we explain the basic construction of static and dynamic hash-based load balancing algorithms and propose a classification of existing and new algorithms.

A. Static and Dynamic Link Selector Functions for Hash-Based Load Balancing

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 traffic distribution of the traffic aggregate $\mathcal{F}(r,d)$ over the links in $\mathcal{L}(r,d)$. Target load fractions other than even load distributions 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 $0, \ldots, |\mathcal{B}(r,d)|-1$. 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 1. The link-specific bin set $\mathcal{B}(r,d,l)$ contains all bins of $\mathcal{B}(r,d)$ with pointers to l.

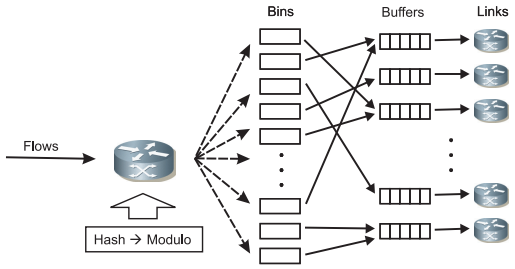


Fig. 1. Data structure of a table-based link selector function.

The assignment between bins and links is fixed for static link selector functions. However, increasing the link-specific bin set $\mathcal{B}(r,d,l)$ 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 perform these actions during runtime and, thus, adapt their dynamic link selector functions to the current load conditions. They check the current load difference $cLD(r,d,l) = cLF(r,d,l) - tLF(r,d,l)$ 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. 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.

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 $cLF(r,d,b) = \frac{cTR(r,d,b)}{\sum_{b' \in \mathcal{B}(r,d)} cTR(r,d,b')}$.

B. Bin Reassignment Algorithms for Dynamic Load Balancing

We define and classify several bin reassignment algorithms for dynamic load balancing in a modular way. We first propose strategies to disconnect bins from link-specific bin sets. Then we suggest methods to add the free bins to new link-specific bin sets. The combination of these strategies yields various bin reassignment algorithms. All algorithms operate on router- and destination-specific structures indexed by a specific (r,d)-tuple.

1) Bin Disconnection Strategies: Bin disconnection strategies differ with regard to the number of simultaneously disconnected links, i.e., either only a single bin is disconnected at once or multiple bins may be disconnected. Furthermore, disconnection strategies may be progressive (-), i.e., they try to bring overloaded links into underload; or they may be conservative (+), i.e., they try to avoid to bring overloaded links into underload.

a) Conservative Single Bin Disconnection (SBD^+): The conservative algorithm SBD^+ disconnects from the link-specific bin set $\mathcal{B}(r,d,l)$ of the heaviest loaded link l the largest bin that does not turn the link l into underload. Thus, we call SBD^+ conservative. SBD^+ does not disconnect any bin if the disconnection of the smallest bin from the heaviest loaded link l turns the link l into underload.

b) Progressive Single Bin Disconnection (SBD^-): The dynamic load balancing algorithm in [12] proposed for best accuracy disconnects the largest bin from the link-specific bin set $\mathcal{B}(r,d,l)$ of the heaviest overloaded link l. It is irrelevant, whether the considered link l turns into underload or not. Therefore, this strategy is progressive and we denote it by SBD^-.

We focus on simple algorithms here. Note that many other methods can be defined for single bin disconnection.

c) Conservative Multiple Bin Disconnection (MBD^+): The conservative multiple bin disconnection strategy disconnects from all overloaded links so many bins until any further removal turns them into underload. The bins within the link-specific sets $\mathcal{B}(r,d,l)$ are checked in a simple greedy manner in the order of decreasing size, whether they turn the link into

underload. If not, the bin is removed. The free bins are stored in a so-called bin pool $\mathcal{BP}(r,d)$.

d) Progressive Multiple Bin Disconnection (MBD^-): The progressive multiple bin disconnection strategy MBD^- works like MBD^+ but it eventually turns each still overloaded link l into underload by removing the smallest bin from its link-specific bin set $\mathcal{B}(r,d,l)$.

2) Bin Reconnection Strategies: A single bin or multiple bins have been disconnected from their link-specific bin sets $\mathcal{B}(r,d,l)$ and must be reconnected to new ones in such a way that the target load fraction $tLF(r,d,l)$ of each link is met. Usually, this objective can be met only approximately. The resulting load balancing inaccuracy on any link l may be measured by the current load difference $cLD(r,d,l)$. As an alternative, this difference may be viewed in a relative way by the relative current load difference $rCLD(r,d,l) = \frac{cLD(r,d,l)}{tLF(r,d,l)}$. Thus, the reconnection of the bins should be optimized with regard to one of these measures. The exact optimization of this problem is difficult given the time constraints in high speed routing. Therefore, we solve it again by simple greedy approaches. All disconnected bins are collected in the bin pool $\mathcal{BP}(r,d)$ where they are sorted according to their size. We select bins in the order of decreasing size for reconnection to the bin sets and propose two simple strategies for this purpose.

a) Absolute Difference Bin Reconnection (ADBR): We reconnect the bin to the link l with the lowest current load difference $cLD(r,d,l)$, i.e. with the largest underload.

b) Relative Difference Bin Reconnection (RDBR): In a first step, we try to reconnect the bin to the link l with the largest underload like above, but only if the bin b does not turn the link into overload. This can be achieved if $\exists l \in \mathcal{L}(r,d) : cLF(r,d,l) + cLF(r,d,b) \leq tLF(r,d,l)$ holds. If this fails, we reconnect the bin b in a second step to a link l that obtains the lowest relative overload among all links in $\mathcal{L}(r,d)$ if the bin b is assigned to its link set $\mathcal{B}(r,d,l)$. Such a link can be formally described by $argmin_{l \in \mathcal{L}(r,d)}(\frac{cLD(r,d,l)+cLF(r,d,b)}{tLF(r,d,l)})$.

Note that many other bin reconnection strategies may be defined.

3) Composition of Bin Reassignment Algorithms: In Section III-B.1 we have proposed several methods for the disconnection of bins. We define the generic algorithm $BinDisconnection$ that may be instantiated by any of the presented options SBD^-, SBD^+, MBD^+, and MBD^-. After their disconnection, the bins are collected in a bin pool $\mathcal{BP}(r,d)$. In Section III-B.2 we have suggested several methods for the reconnection of these bins to new link-specific bin sets. We define the generic algorithm $BinReconnection$ that may be instantiated by either $ADBR$ or $RDBR$. Thus, we get 8 substantially different bin reassignment methods by the generic approach presented by Algorithm 1.

In the following, we use a slash-notation to refer to the actual algorithms, e.g. $SBD^-/ADBR$. This is the algorithm proposed by [12], while the other 7 combinations are new methods. Many other options may also be implemented. Again, we underline that these are all greedy algorithms which are only heuristics and achieve certainly not the optimum.

$\mathcal{BP}(r,d) = BinDisconnection(\{\mathcal{B}(r,d,l) : l \in \mathcal{L}(r,d) \wedge cLD(r,d,l) > 0\})$
while $\mathcal{BP}(r,d) \neq \emptyset$ **do**
  $b_{max} = argmax_{b \in \mathcal{BP}(r,d)}(cLF(r,d,b))$
  $BinReconnect(\{\mathcal{B}(r,d,l) : l \in \mathcal{L}(r,d)\}, b)$
end while

Algorithm 1: Generic bin reassignment method.

However, simplicity and fast execution of the algorithms count more than optimality. This was one of our design goals.

IV. Simulation Results

In this section we provide extensive simulation results that help to understand the accuracy and the flow reassignment behavior of dynamic load balancing algorithms under various conditions. First, we explain our simulation methodology. Then, we investigate the influence of flows with heterogeneous flow rates and the offered load for static hashing. We show that the distribution accuracy can be improved by simple dynamic load balancing and that a larger number of bins leads to better results. We compare several dynamic load balancing algorithms and investigate the impact of the length of the bin reassignment interval on the flow reassignment rate and the load balancing accuracy.

A. Simulation Methodology

We simulate on a very small simulation topology on the flow level. We use synthetic flow IDs instead of packet traces and generate the flows according to a Poisson model. We motivate these assumptions in the following.

1) Simulation Topology: We are interested in the load balancing behavior for a flow set $\mathcal{F}(r,d)$ at router r and destined for destination d. Therefore, we simulate only 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 following, we fix the parameters r and d and abandon them from our notation for easier readability. This is possible as we consider only a single router and a single destination.

2) Flow Level Simulation: Many related studies 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 focus on the flow reassignment rate λ_{FR}. It is affected only by the dynamic load balancing and influences the packet reordering probability proportionally. In addition, flow level simulations run much faster and allow us to produce very reliable results.

3) Synthetic Flow ID Generation: In many studies real traffic traces are used to evaluate the quality of load balancing mechanisms and to emphasize that the results are realistic. This is important to assess the quality of hash functions for a certain application. In our study, we use the 16-bit CRC

function because the authors of [13] have shown that "hashing using a 16-bit CRC over the five-tuple gives excellent load balancing performance". We are interested in the general potential of static and different dynamic load balancing algorithms and not in the quality of different hash functions. Therefore, we use synthetically generated flow IDs to avoid any correlation effects within a specific trace.

4) Traffic Model: The interarrival time of flows on Internet links are exponentially distributed with rate λ_{IAT} [25]–[27]. Therefore, the Poisson model is well applicable on the flow level. The call holding times are identically and independently distributed with a mean value of $E[B] = 90$ s. Thus, the offered load can be calculated by $a = \lambda_{IAT} \cdot E[B]$ measured by the pseudo unit Erlang (Erl) and can be viewed as the average number of simultaneous flows. The variation of the rates of different flows has a significant impact on the quality of the load balancing mechanisms [10]. In fact, there are a few large flows (elephants) producing fifty to sixty percent of the total traffic while the rest is due to many small flows (mice) [28], [29]. As a consequence, our traffic model is multi-rate to capture this effect. We use $n_r = 3$ different flow types $r_i, 0 \leq i < n_r$ with flow rates $c(r_i) \in \{64, 256, 2048\}$ kbit/s. Details can be found in [30].

5) Performance Measures and Simulation Credibility: We consider two important performance aspects for load balancing algorithms: load balancing accuracy and the flow reassignment rate. We measure the load balancing accuracy of the current load fraction $cLF(l)$ for each link $l \in \mathcal{L}$ and capture a time-weighted histogram for $cLF(l)$. We define the absolute deviation of the load fractions $cLF(l)$ from their target values $tLF(l)$ averaged over all links $l \in \mathcal{L}$ as as an additional performance measure for the load balancing inaccuracy

$$I = \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} |cLF(l) - tLF(l)| = \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} |cLD(l)| \qquad (1)$$

and use its mean $E[I]$ to capture the inaccuracy by a single number.

We calculated confidence intervals for all performance metrics used in this work based on standard simulation techniques such as replicate-delete [31]. We simulated so long that the 99% confidence intervals deviate at most 1% from the respective mean values. Thus, they are very small which proves the statistical significance of our results. As they are hardly visible, we do not show them in the following figures.

B. Impact of Exogenous Parameters on the Accuracy of Static Load Balancing

In our first experiments, we study the influence of the flow rate variability and the offered load on the load balancing accuracy of static load balancing. We assume only two outgoing links to which the traffic should be forwarded equally.

First, we assume an offered link load of 100 Erl and consider the influence of the flow rate variability. In the case of homogenous flows, all flows have a rate of 256 kbit/s whereas in the case of heterogenous flows, the flows have only 64 and 2048 kbit/s but the same mean of 256 kbit/s, which yields a coefficient of variation of 2.29. The results for the current load fraction $cLF(l)$ captured in the time-weighted histogram can be transformed into the distribution function from Figure 2, which makes it easy to differentiate curves from several experiments. The x-axis shows the load fraction on the first link in percent with a granularity of 1%. 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 result for the second link is symmetric as we consider two links here. The load balancing accuracy is high if the curve increases around the respective target load fraction $tLF(l)$ with a steep slope. The probability function for homogeneous flows illustrates that the measured load fraction varies from 0.35 to 0.65 in spite of a target load fraction of 0.5. The probability follows exactly a binomial distribution according to $P(cLF(l) = i) = \binom{100}{i} 0.5^{100}$ as load balancing reduces to the task of distributing the number of currently active flows over paths. In case of heterogeneous flows, the flatter curve shows that the load balancing accuracy is significantly decreased and the average inaccuracy $E[I]$ increases from 3.90% to 10.06%. Thus, the flow rate variability has a clear impact on the load balancing accuracy. This finding is in line with the results in [10]. We conduct all following studies with heterogeneous flows because this model is more realistic.

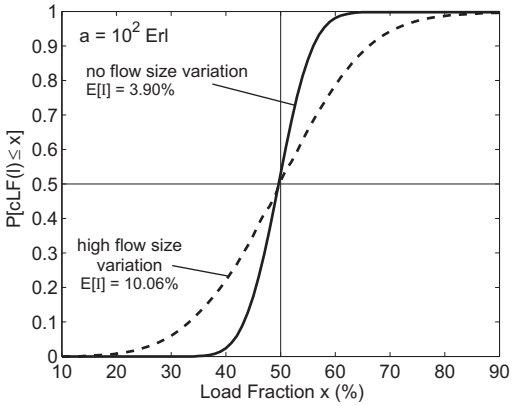


Fig. 2. Impact of flow rate variability on the traffic distribution for static load balancing.

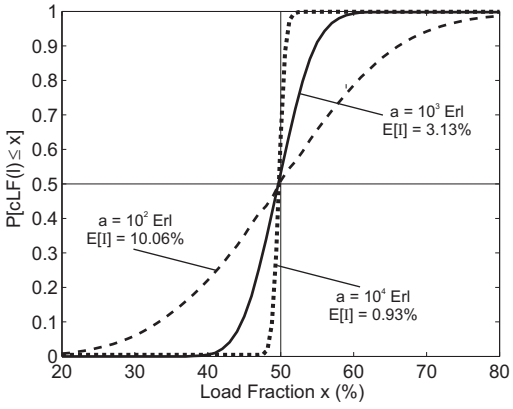


Fig. 3. Impact of the offered load on the traffic distribution for static load balancing.

In the next experiment, we consider the number of simultaneous flows in the flow aggregate $\mathcal{F}$. Figure 3 shows the load balancing accuracy for 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. The data in the figure are given for heterogeneous flows. For homogeneous flows this result can be easily derived from the binomial distribution above: the coefficient of variation of the load fraction can be calculated by $c_{var} = \frac{1}{\sqrt{a}}$. An offered load of 10 Erl is definitely too small for load balancing since we observed almost any load fractions between 0 and 1 and, thus, is not shown here. For 10^3 Erl we get better observations between 0.38 and 0.62 for static load balancing algorithms and an average inaccuracy of 3.13% instead of 10.06% as observed for $a = 10^2$ Erl. Very high traffic aggregates at $a = 10^4$ Erl lead to almost perfect load balancing with a very low mean inaccuracy of 0.93%. In the following experiments, we consider an offered load of 10^2 Erl because it is a moderate aggregation level and, thereby, more challenging for the load balancing accuracy.

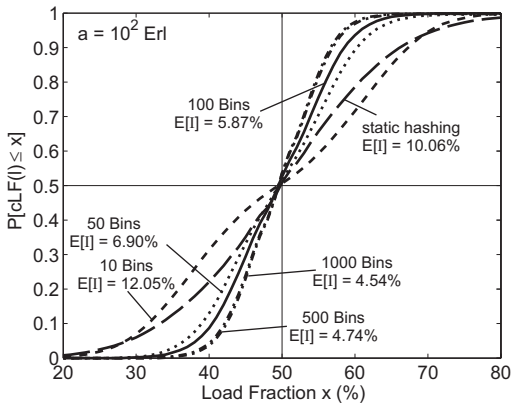
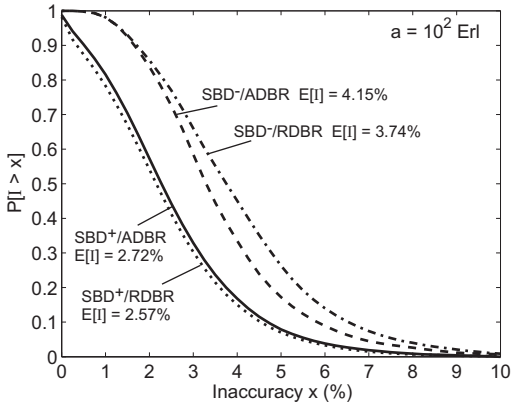


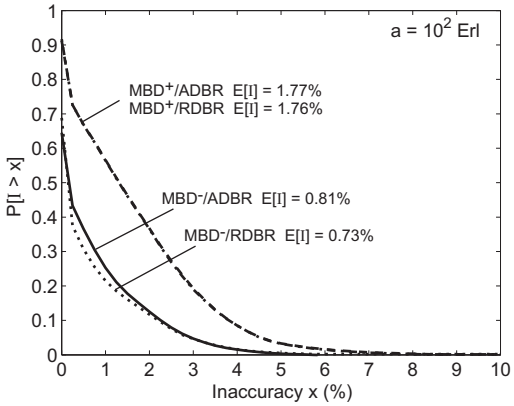
Fig. 4. Impact of the number of bins on the traffic distribution for $SBD^-/ADBR$ dynamic load balancing.

C. Accuracy Increase through Dynamic Load Balancing

Now we consider possible performance gains through dynamic load balancing algorithms and first analyze $SBD^-/ADBR$ as it has been proposed in [12]. We use a bin reassignment interval with a length of $t_r = 1$ s. The size of the bin set $\mathcal{B}$ is crucial for the accuracy of table-based load balancing. Figure 4 shows the distribution function of the load fraction for static load balancing and for dynamic load balancing with a different number of bins in the two-link experiment from above. With only 10 bins, the average load balancing inaccuracy is with 12.05% larger than with static load balancing (10.06%). The small number of bins with dynamic adaptation is counterproductive. However, there is a significant improvement of the inaccuracy for 50 bins (6.90%), 100 bins (5.87%), and 500 bins (4.74%). Another doubling of the number of bins does not lead to any clear performance gain (4.54%). The algorithms become more complex if the number of bins increases. In the following, we work with 100 bins because they lead to a sufficiently high accuracy and impose still moderate complexity, which is important for technical feasibility.



(a) The family of SBD-based algorithms.



(b) The family of MBD-based algorithms.

Fig. 5. Complementary distribution function of the load balancing inaccuracy for various load balancing algorithms.

D. Comparison of the Accuracy of Different Dynamic Load Balancing Algorithms

We have seen from the two-link experiments above that dynamic load balancing can significantly increase the load balancing accuracy. We compare now different dynamic load balancing algorithms and use a more sophisticated experiment for that purpose. The traffic is distributed over four links with target load fractions of 10%, 20%, 30%, and 40%. We first study the inaccuracy of the single bin disconnection (SBD) and multiple bin disconnection (MBD) algorithm families. Then, we show the details on each of the four links for SBD to motivate the observed performance. And finally, we contrast the detailed results for the best MBD algorithm to the SBD results to illustrate the potential of multiple bin disconnection.

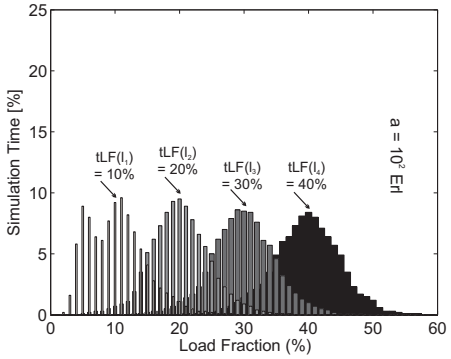
1) Comparison of the Inaccuracy Distribution: The inaccuracy I defined in Equation 1 captures the histograms for the current load fractions $cLF(l)$ as shown in Figure 6(a) on each of the four links in a single measure and enables us to compare several algorithms more easily. Figure 5(a) illustrates the complementary distribution function of the inaccuracy I for the entire SBD algorithm family. The x-axis shows the

inaccuracy in percent with a granularity of 1% and the y-axis shows the probability that the observed inaccuracy is larger than the value x at any time instant. The faster the curves decay, the better is the load balancing accuracy. The SBD^+-based algorithms ($E[I] = 2.57\%$ and 2.72%) are significantly more accurate than the SBD^--based algorithms ($E[I] = 3.74\%$ and 4.15%). For SBD^-, the version based on relative difference bin reassignment ($RBDR$) is significantly more inaccurate than the version based on absolute difference bin reassignment ($ABDR$) while there is hardly any difference between them for SBD^+.

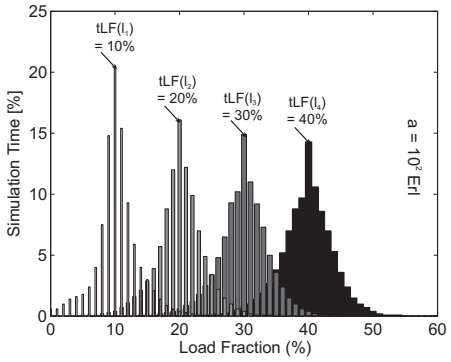
The MBD algorithm family outperforms the SBD family clearly as illustrated with the corresponding results for the MBD algorithms in Figure 5(b). The lines decay much faster. Here, the MBD^- versions ($E[I] = 0.73\%$ and 0.81%) are significantly more accurate than the MBD^+-based methods ($E[I] = 1.76\%$ and 1.77%). For MBD^-, the $RDBR$-based version is only little more inaccurate than the $ABDR$-based approach and for MBD^+ we cannot see any difference between them.

2) Comparison of SBD-Based Load Balancing Algorithms: Figures 6(a) to 6(c) show the histograms of the load fraction on each of the four links for various SBD-based algorithms to understand the above results for the inaccuracy in detail. For the $SBD^-/ADBR$ method in Figure 6(a), the deviations around the target load fraction is symmetric and similar for all links except for the one with the smallest target load fraction. This phenomenon is due to the flow size variation. Generally, the range of observed load fractions is still quite broad for $SBD^-/ABDR$. It removes always the largest bin from the link with the heaviest overload. This bin may be too large to balance the load and its disconnection causes significant underload on the considered link. In addition, this may cause oscillations if the same bin is exchanged back and forth between the same two links. As illustrated in Figure 6(b), the conservative algorithm $SBD^+/ABDR$ avoids this problem, leads to more accurate load balancing and clearly outperforms the progressive approach $SBD^-/ADBR$. It is interesting that links with a smaller load fraction have a larger peak around their target load fraction, which is a good feature. This effect is enforced by the $SBD^+/RDBR$ approach seen in Figure 6(c) as it tries to minimize the load deviation relative to the respective target value. However, the data reveal that the impact of the $RDBR$ strategy is quite weak. The improvement of the load balancing accuracy for links with a low target load fraction is reached at the expense of a slightly degraded load balancing accuracy for links with a high target load fraction. The same phenomenon can be observed with $SBD^-/ADBR$ and $SBD^-/RDBR$.

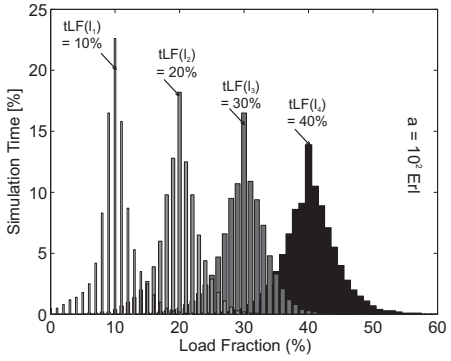
3) Potential of MBD-Based Load Balancing Algorithms: Figure 6(d) illustrates the load balancing accuracy for $MBD^-/ADBR$. It is significantly better compared to the SBD-based methods and to emphasize this we draw particular attention to the differently scaled y-axis. This clearly shows the benefit of MBD opposed to SBD.



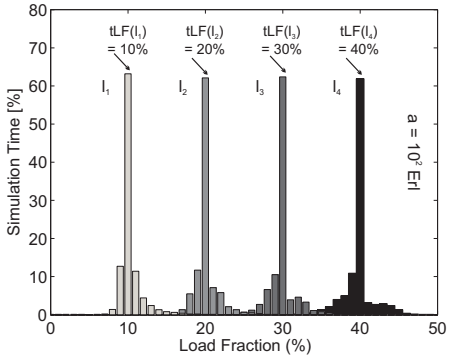
(a) $SBD^-/ADBR$



(b) $SBD^+/ADBR$

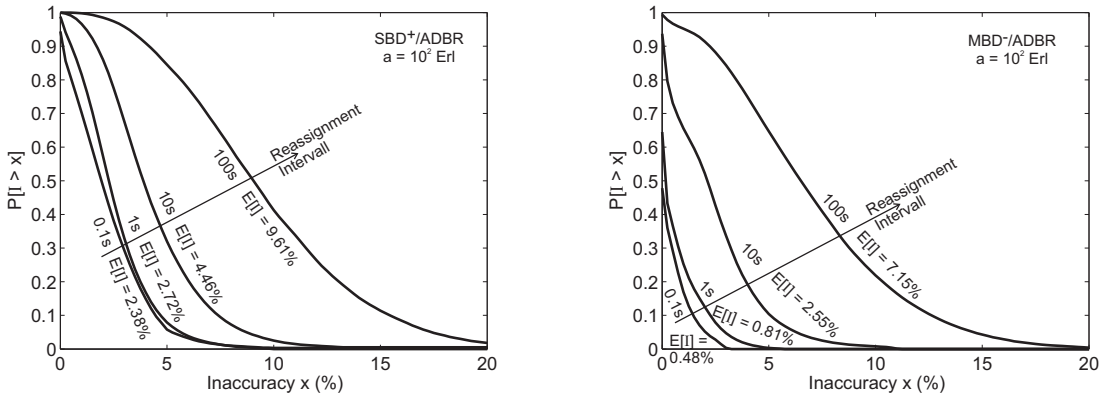


(c) $SBD^+/RDBR$



(d) $MBD^-/ADBR$

Fig. 6. Accuracy of load balancing over four links for various algorithms.



(a) $SBD^+/ADBR$. (b) $MBD^-/ADBR$.

Fig. 7. Impact of the bin reassignment interval t_r on the load balancing accuracy.

The accuracy is quite similar for each link. However, links with small target load fractions rather tend to have positive load deviations while links with large target load fractions rather tend to have negative load deviations. The details for the other MBD versions are omitted here.

In the following we use $ADBR$-based algorithms because they are less complex and more accurate.

E. Impact of the Bin Reassignment Interval Length on the Accuracy and the Flow Reassignment Rate

The duration of the flows and the application frequency of dynamic reassignment steps have a significant impact on the load balanced results. In our simulations, the flow durations are exponentially distributed with a mean value of 90 s but we do not further elaborate on this issue since this is not a parameter under control. We rather investigate the load balancing accuracy depending on the reassignment interval length t_r.

Figure 7(a) shows the impact of t_r on the load balancing accuracy for $SBD^+/ADBR$. The complementary distribution functions of the inaccuracy are similar for $t_r = 0.1$ s and $t_r = 1$ s with mean values of $E[I] = 2.38\%$ and 2.72%. The accuracy is clearly degraded for $t_r = 10$ s ($E[I] = 4.46\%$) and it is not acceptable for $t_r = 100$ s ($E[I] = 9.61\%$). We get similar results for $MBD^-/ADBR$ in Figure 7(b). The inaccuracy for $t_r = 100$ s is not acceptable ($E[I] = 7.15\%$) but the inaccuracy for $t_r = 10$ s ($E[I] = 2.55\%$) is comparable to the best accuracy of $SBD^+/ADBR$. The accuracy for $t_r = 0.1$ s and $t_r = 1$ s are also similar, but with $E[I] = 0.48\%$ and 0.81% it is significantly better than for the corresponding values of $SBD^+/ADBR$.

The flow reassignment rate λ_{FR} is the average number of reassignments of a flow per second. If we multiply λ_{FR} with the lifetime of a given flow we get the number of reassignments this flow will perceive over its complete duration on average. The length of the bin reassignment interval t_r has a significant impact on the rate λ_{FR}. Figure 8 compiles the flow reassignment rates for $SBD^+/ADBR$ and $MBD^-/ADBR$. The flow reassignment rate increases for both algorithms by a factor of 10 if t_r decreases by a factor of 10 from 100 s to 10 s. We conclude that the same number of flows is reassigned whenever the load is balanced for $t_r \in \{10, 100\}$ s. A further reduction of t_r increases the reassignment rate significantly less. Hence, the number of reassigned flows within one step decreases.

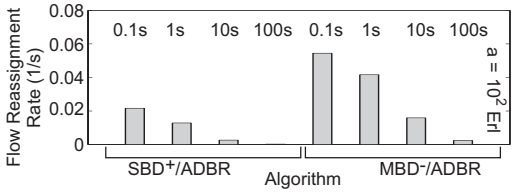


Fig. 8. Impact of the bin reassignment interval t_r on the flow reassignment rate.

$SBD^+/ADBR$ and $MBD^-/ADBR$ achieve good load balancing results for $t_r = 0.1$ s and $t_r = 1$ s. However, for $t_r = 0.1$ s the flow reassignment rate is much higher. A similar accuracy can be obtained for $MBD^-/ADBR$ at $t_r = 10$ s and $SBD^+/ADBR$ at $t_r = 1$ s with about the same flow reassignment rate. After all, a bin reassignment interval length of 1 s should be chosen for both algorithms. Then, the flow reassignment rate is about 0.04 $\frac{1}{\text{s}}$ for $MBD^-/ADBR$ which means that a flow is reassigned every 25 s and that a flow with a duration of 90 s is reassigned 3.6 times on average. Note that packet reordering occurs less frequently because packets do not get necessarily out of order when flows are switched to another paths. The flow reassignment rate may be further reduced for MBD algorithms if the reconnection process tries to reconnect bins to their previous links if possible. This obviously already happens by chance but more intelligent algorithms can enforce this. Their complexity may be feasible for a bin reassignment interval length of $t_r = 1$ s such that this gives room for further research.

V. Conclusion

In this paper we investigated the load balancing accuracy and the flow reassignment rate for various load balancing algorithms in different networking scenarios. First, we gave a broad overview on the use of load balancing algorithms but in our study we focused on load balancing for multipath Internet routing. We presented a taxonomy of load balancing algorithms that contained existing and new methods.

Then, we showed by means of simulation that the load balancing accuracy for static load balancing algorithms depends on the flow rate variability and the offered load. Dynamic load balancing algorithms, that use indirect table-based hashing over intermediate bins, can improve the load balancing accuracy, in particular, if the number of bins is large enough. We showed that the deviation from the target load distribution may be significantly different for various load balancing algorithms. The distribution accuracy improves significantly if more than a single bin may be reassigned in a single load balancing step. Our newly proposed load balancing approaches clearly outperform existing solutions regarding the load balancing accuracy. We showed that a bin reassignment interval of 1 s is good enough to achieve a good accuracy and that flows are reassigned every 25 s to other paths which may cause packet reordering. Future enhancements to our algorithms may reduce this rate.

In this work, we considered only the load balancing at a single router. In networks, traffic may be balanced at different stages and the success of load balancing in an interior router may be influenced by the load balancing actions performed by its predecessors. In addition, it is not clear, how the underlying hashing algorithms must be designed to provide good load distribution in entire networks. Currently, we are working on these issues.

Acknowledgment

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

References

[1] G. Schollmeier, J. Charzinski, A. Kirstädter, C. Reichert, K. J. Schrodi, Y. Glickman, and C. Winkler, "Improving the Resilience in IP Networks," in *IEEE High Performance Switching and Routing (HPSR)*, Torino, Italy, June 2003.

[2] 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.

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

[4] *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).

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

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

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

[8] 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.

[9] 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.

[10] W. Shi, M. H. MacGregor, and P. Gburzynski, "An Adaptive Load Balancer for Multiprocessor Routers," in *SPECTS 2004*, San Jose, CA, July 2004, pp. 671–679.

[11] G. Dittmann and A. Herkersdorf, "Network Processor Load Balancing for High–Speed Links," in *SPECTS 2002*, San Diego, CA, 2002, pp. 727–735.

[12] T. W. Chim and K. L. Yeung, "Traffic Distribution over Equal-Cost-Multi-Paths," in *Proceedings of IEEE International Conference on Communications (ICC)*, Paris, France, June 2004.

[13] 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.

[14] I. Gojmerac, T. Ziegler, F. Ricciato, and P. Reichl, "Adaptive Multipath Routing for Dynamic Traffic Engineering," in *IEEE Globecom*, San Francisco, Nov 2003.

[15] C. Hoogendoorn, K. Schrodi, M. Huber, C. Winkler, and J. Charzinski, "Towards Carrier-Grade Next Generation Networks," in *International Conference on Communication Technology (ICCT)*, Beijing, China, April 2003.

[16] M. Menth, J. Milbrandt, and S. Kopf, "Impact of Routing and Traffic Distribution on the Performance of Network Admission Control," in 9^{th}*IEEE Symposium on Computers and Communications (ISCC)*, Alexandria, Egypt, June 2004, pp. 883 – 890.

[17] A. Zinin, *Cisco IP Routing, Packet Forwarding and Intra-domain Routing Protocols*. Addison Wesley, 2002, ch. 5.5.1.

[18] *Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures*, International Organization for Standardization, ISO/IEC 13239:2002.

[19] *Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion*, International Telecommunication Union – Telecom Standardization (ITU-T), Recommendation V.43 (03/02).

[20] Ross, Williams N., "A Painless Guide to CRC Error Detection Algorithms," http://www.ross.net/crc/, May 1996.

[21] P. H. Fredette, "The Past, Present, and Future of Inverse Multiplexing," *IEEE Communications Magazine*, vol. 32, pp. 42–46, April 1994.

[22] H. Adiseshu, G. Parulkar, and G. Varghese, "A Reliable and Scalable Striping Protocol," in *Proceedings of ACM SIGCOMM Computer Communication Review*, vol. 26, August 1996.

[23] D. G. Thaler and C. V. Ravishankar, "Using Name-Based Mappings to Increase Hit Rates," *IEEE/ACM Transactions on Networking*, vol. 6, no. 1, 1998.

[24] R. Martin and M. Menth, "Improving the Timeliness of Rate Measurements," in 12^{th}*GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB) together with* 3^{rd}*Polish-German Teletraffic Symposium (PGTS)*, Dresden, Germany, Sept. 2004, pp. 145 – 154.

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

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

[27] 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.

[28] 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.

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

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

[31] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3^{rd} ed. McGraw-Hill, 2000.