Dynamic Load Balancing Without Packet Reordering

Srikanth Kandula, MIT, kandula@mit.edu
Dina Katabi, MIT, dk@mit.edu
Shantanu Sinha, Microsoft, shans@microsoft.com
Arthur Berger, MIT/Akamai, awberger@mit.edu

ABSTRACT

Dynamic load balancing is a popular recent technique that protects ISP networks from sudden congestion caused by load spikes or link failures. Dynamic load balancing protocols, however, require schemes for splitting traffic across multiple paths at a fine granularity. Current splitting schemes present a tussle between slicing granularity and packet reordering. Splitting traffic at the granularity of packets quickly and accurately assigns the desired traffic share to each path, but can reorder packets within a TCP flow, confusing TCP congestion control. Splitting traffic at the granularity of a flow avoids packet reordering but may overshoot the desired shares by up to 60% in dynamic environments, resulting in low end-to-end network goodput.

Contrary to popular belief, we show that one can systematically split a single flow across multiple paths without causing packet reordering. We propose FLARE, a new traffic splitting algorithm that operates on bursts of packets, carefully chosen to avoid reordering. Using a combination of analysis and trace-driven simulations, we show that FLARE attains accuracy and responsiveness comparable to packet switching without reordering packets. FLARE is simple and can be implemented with a few KB of router state.

Categories and Subject Descriptors

C.2.2 [**Communication Networks**]: Network Protocols

General Terms

Algorithms, Design, Performance, Theory

Keywords

FLARE, Packet Reordering, Traffic Splitting

1. INTRODUCTION

Load balancing is common in ISP networks. It is a key component of traffic engineering, link bundling, and equal cost multi-path routing [8, 28, 14, 6, 29]. Recent trends in load balancing point toward dynamic protocols. These protocols map the traffic of an ingress-egress router pair onto multiple paths and adapt the share of each path in real-time to avoid hot-spots and cope with failures [18, 30, 12]. Protocols like TeXCP [18], COPE [30], and MATE [12] have demonstrated the value of such approaches in reducing costs and increasing network robustness.

Dynamic load balancing needs schemes that split traffic across multiple paths at a fine granularity. Current traffic splitting schemes, however, exhibit a tussle between the granularity at which they partition the traffic and their ability to avoid packet reordering. Packet-based splitting quickly assigns the desired load share to each path. When paths differ in delay, however, splitting at packet granularity can reorder a large number of packets. TCP confuses this reordering as a sign of congestion, resulting in degraded performance. Even some UDP-based applications such as VoIP [20] are sensitive to packet reordering. Flow-based splitting, on the other hand, pins each flow to a specific path and avoids packet reordering. But, flows differ widely in their sizes and rates, and once assigned, a flow persists on the path throughout its lifetime [34, 22, 26]. Consequently, flow-based splitting may assign inaccurate amounts of traffic to each path or fail to quickly re-balance the load in the face of changing demands. This inability to quickly react to traffic spikes congests links and reduces network goodput.

This paper shows that one can obtain the accuracy and responsiveness of packet-based splitting and still avoid packet reordering. We introduce FLARE,[1] a new traffic splitting algorithm. FLARE exploits a simple observation. Consider load balancing traffic over a set of parallel paths (Fig. 1). If the time between two successive packets is larger than the maximum delay difference between the parallel paths, one can route the second packet —and subsequent packets from this flow— on any available path with no threat of reordering. Thus, instead of switching packets or flows, FLARE switches packet bursts, called flowlets. By definition, flowlets are spaced by a minimum interval δ, chosen to be larger than the delay difference between the parallel paths under consideration. FLARE measures the delay on these paths and sets the flowlet timeout, δ, to their maximum delay difference. The small size of flowlets lets FLARE split traffic dynamically and accurately, while the constraint imposed on their spacing ensures that no packets are reordered.

This paper makes the following contributions.

(a) It introduces FLARE, showing that it is possible to systematically slice a TCP flow across multiple paths without causing packet reordering.

(b) It formally analyses the traffic splitting problem. Our analysis shows that deviation from the desired traffic split is always smaller with flowlets than with flows, and depends on the number of flowlets per time unit and the coefficient of variation (standard deviation/mean) of flowlet sizes.

(c) It evaluates FLARE using extensive simulations run on 863 traces from tier-1 and regional providers, and reveals the following findings.

- FLARE's bandwidth allocation is highly accurate. It is always within a few percent of the desired shares, for both static and dynamic splits. In contrast, splitting schemes that pin a flow to a path like flow-based, S-Hash and BIN-based overshoot the desired bandwidth allocation by an average of 20%-60%, in dynamic environments, which reduces the overall goodput under high load. Packet-based splitting, on the other hand, causes 1%-3% extra 3 duplicate TCP acks, and thus hurts end-to-end goodput.
- FLARE is remarkably robust to errors in estimating the

[1]Flowlet Aware Routing Engine

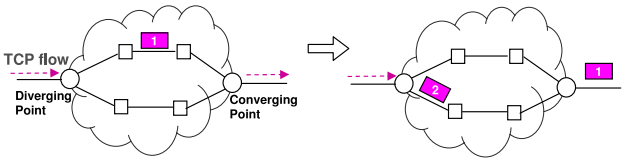


Figure 1: As long as the inter-packet spacing is larger than the delay difference between the two paths, one can assign the two packets to different paths without risking packet reordering.

delay difference of the parallel paths. It operates correctly even with an estimation error as high as 100ms, a value larger than the coast-to-coast round trip delay.

- FLARE's state overhead is small. On all of the 863 traces examined (with link capacities upto 10Gbps), FLARE can track flowlets with a hash table of approximately 1000 entries, which fits in a few KB of memory. Further, the state required scales roughly linearly with link capacity.

With its responsiveness and robustness against packet reordering, FLARE enables a new generation of dynamic load balancing schemes such as TeXCP [18] and COPE [30] to deliver its potential. Further, its simplicity and low-overhead make it easy to implement and use in routers.

2. RELATED WORK

The work closest to FLARE is in the context of circuit and optical switching It observes that switching user-data bursts allows circuits to be established once for each burst, and a single routing table look-up to be made for the entire burst [4]. FLARE's novelty stems from switching bursts to achieve responsive load balancing without packet reordering.

In datagram networks, a few proposals for traffic splitting forward packets onto multiple paths using a form of weighted round-robin or deficit round robin [27] scheduling. These schemes cause significant packet reordering and thus are not used in practice. Alternative schemes avoid packet reordering by consistently mapping packets from the same flow to the same path. Commercial routers [11, 17] implement the Equal-Cost Multipath (ECMP) feature of routing protocols such as OSPF and IS-IS. Hash-based versions of ECMP divide the hash space into equal-size partitions corresponding to the outbound paths, hash packets based on their endpoint information, and forward them on the path whose boundaries envelop the packet's hash value [9, 29]. Though these schemes provide a good performance when operating with static load balancing, they are unsuitable for the emergent dynamic load balancing protocols [18, 30], where they may overshoot the desired load by as much as 60% (as shown in §9.5).

A few papers analyze the performance of various splitting schemes. Cao et al. [9] evaluate the performance of a number of hashing functions used in traffic splitting. Rost and Balakrishnan [25] evaluate different traffic splitting policies, including rate-adaptive splitting methods.

Related work also exists in the area of load balancing [18, 12, 29, 31, 13]. Load balancing needs to split traffic across multiple paths. Recent proposals, such as TeXCP [18], COPE [30] and MATE [12], focus on dynamic approaches where the desired split adapts to changes in the observed load. This form of dynamic load balancing stresses current traffic splitting schemes, which cannot quickly adapt the amount of traffic on each path. FLARE provides an ideal traffic splitting scheme for these dynamic load balancing algorithms because it quickly and accurately matches the desired traffic split, while limiting packet reordering.

Of relevance to our work are recent efforts on improving TCP's robustness to packet reordering [21, 7, 33, 1]. If the future TCP would be completely immune to packet reordering, a packet-based splitting technique would outperform all other approaches. However, there is no indication such a TCP will be deployed in the near future. Until then, there is a need for an accurate traffic splitting mechanism with minimal reordering.

3. SPLITTING PROBLEM

The traffic splitting problem is formalized as follows [25]. The aggregate traffic arriving at a router, at rate R, is composed of a number of distinct transport-layer flows of varying rates. Given N disjoint paths that can be used concurrently, and a split vector $\vec{F} = (F_1, F_2, ..., F_N)$, where $F_i \in [0, 1]$ and $\sum_{i=1}^{i=N} F_i = 1$, find a packet-to-path assignment such that the traffic rate on path i is equal to $F_i \times R$.

The splitting problem is a key component of the general problem of load balancing. In addition to a traffic splitter, balancing the load across multiple paths requires a mechanism to find the splitting vector $\vec{F}$. The splitting vector may be static as in the case of balancing the load across a link bundle connecting two routers. It might also dynamically adapt to the congestion state along each path. This paper assumes that the splitting vector is either statically configured or dynamically provided by one of the recently proposed traffic engineering solutions [18, 29, 12, 30].

4. FLOWLET SWITCHING

This section proposes *flowlet-based splitting* which aims to combine the responsiveness of packet-based splitting with the ability of flow-based splitting to avoid packet reordering.

A flowlet is a burst of packets from a given transport layer flow. Flowlets are characterized by a timeout value, δ, which is the minimum inter-flowlet spacing, i.e., packet spacing within a flowlet is smaller than δ.

Flowlet-based splitting exploits a simple observation. Consider the scenario in Fig. 1 where a set of parallel paths diverge at a particular point and converge later. Each path contains some number of hops. Given two consecutive packets in a flow, if the first packet leaves the convergence point before the second packet reaches the divergence point, one can route the second packet —and subsequent packets from this flow— on to any available path with no threat of reordering. Thus, by picking flowlet timeout larger than the maximum latency of the set of parallel paths, consecutive flowlets can be switched independently with no danger of packet reordering. In fact, for any set of parallel paths, we can further tighten the timeout value to the difference between the maximum and minimum path latencies. We call this maximum delay difference, the Minimum Time Before Switch-ability (MTBS). As long as the flowlet timeout, δ, is larger than the MTBS, flowlet switching does not cause packet reordering.

5. FLOWLET AWARE ROUTING ENGINE

FLARE is a traffic splitting mechanism. It resides on a router that feeds multiple parallel paths. It takes as input a split vector that could change over time. Upon receiving a packet, FLARE determines the best path along which to

route the packet to achieve the desired split vector, and forwards the packet appropriately.

FLARE relies on the flowlet abstraction to dynamically and accurately split traffic along multiple paths without causing reordering. FLARE has these three components.

(a) MTBS estimator: FLARE uses periodic pings to estimate the delay difference between the parallel paths across which it splits the traffic. It smoothes the estimate using an exponential running average. FLARE sets the flowlet timeout value δ to the maximum delay difference between the parallel paths. This allows FLARE to assign flowlets of the same flow to different parallel paths without reordering packets. Of course, the delay estimate may contain errors and variations. In §9.2, we use a large and diverse set of trace-driven experiments to show that FLARE is robust to such estimation errors.

(b) Flowlet-to-path assignment: FLARE uses a hash table that maps flowlets to paths. Each table entry contains two fields `last_seen_time` and `path_id`. When a packet arrives, FLARE computes a hash of the source IP, destination IP, source port and destination port. (The authors of [9] recommend a CRC-16 hash.) This hash is used as the key into the flowlet table. If the current time is smaller than `last_seen_time + `δ, the packet is part of an already existing flowlet. In this case, the packet is sent on the path identified by `path_id` and `last_seen_time` is set to current time. Otherwise, the packet begins a new flowlet and may be assigned to a new path. FLARE maintains a token counter for each path to estimate how far the path is from its desired load. It assigns new flowlets to the path with the maximum number of tokens, sets `path_id` to the new path id, and sets `last_seen_time` to current time.

(c) Token-counting algorithm: The token-counting algorithm estimates how far a path is from its desired allocation. FLARE assigns a token counter, t_i, to each path, i, of the set of parallel paths. The counter accumulates credits proportionally to how far the path is from its desired load. In particular, for every packet of size b bytes, all token counters are updated as follows:

$$t_i = t_i + F_i \times b, \qquad \forall i \tag{1}$$

where F_i is the fraction of the load to be sent on path i. The packet is assigned to a path according to the flowlet-to-path assignment algorithm. Once the packet has been assigned to a particular path j, the corresponding token counter is decremented by the size of the packet:

$$t_j = t_j - b. \tag{2}$$

The intuition behind the token counting algorithm is simple. The token counters succinctly track the cumulative effect (deficit) of past assignment decisions – the fewer the bytes a tunnel gets, compared to its desired share, the larger its number of tokens. To correct for this deficit, whenever a new flowlet arrives, it is assigned to the tunnel with the maximum number of remaining tokens. But, it is also undesirable to correct for decisions that have been made a long time in the past. To bound the duration for which history is maintained, FLARE introduces the notion of a *measurement interval* (which defaults to 0.5s). Every measurement interval, the token counters are reset allowing the splitter to get a fresh start.

6. ANALYSIS

We formally analyze flowlet-based splitting to understand the parameters controlling its performance.

6.1 Probabilistic Traffic Assignment

We analyze a splitting algorithm that assigns units of traffic –packets, flowlets, flows– to paths with probabilities proportional to the desired split vector; i.e., assign a traffic unit to path i with probability f_i, where f_i is the desired load share for path i. We are interested in how far the amount of traffic assigned to a path is from its desired load.

We begin by defining some random variables. Let S_j denote the size of the j^{th} unit, and I_j^i be an indicator variable that is 1 if the j^{th} unit is assigned to path i and 0 otherwise. Let $N(t)$ be the number of traffic units to arrive over the interval $(0, t]$. At time t, the deficit of a path is defined as:

$$D_i(t) = \frac{\sum_{i=1}^{N(t)} \left(I_j^i - f_i\right) S_j}{E[\sum_{j=1}^{N(t)} S_j]}.$$

The definition of deficit is intuitive; the numerator captures the difference between the amount of traffic assigned to the path $\sum_j I_j^i S_j$, and its desired load share $\sum_j f_i S_j$. To make the quantity dimensionless, we normalize by the expected volume of traffic to arrive by time t.

THEOREM 6.1. *A path's deficit is bounded from above:*

$$P(|D_i(t)| > \epsilon) \quad < \quad \frac{1}{4\epsilon^2 E[N(t)]} \left(\left(\frac{\sigma_S}{\mu_S}\right)^2 + 1 \right), \tag{3}$$

where σ_S and μ_S are the standard deviation and mean of the traffic unit size, respectively.

Proof A proof is given in Appendix A.

Equation 3 identifies two reasons why flowlet-based splitting should do better than flow-based splitting. It shows that the deviation from desired split depends on two parameters: the number of traffic units $N(t)$, and the coefficient of variation $\frac{\sigma_S}{\mu_S}$. Since any flow will generate at least one flowlet, and in most cases many, $N(t)$ is significantly larger for flowlets than flows, making deviations from desired load less likely under flowlet-based splitting. Indeed, our traces show that $N(t)$ is more than an order of magnitude larger for flowlets than for flows (see Fig. 3). As for the coefficient of variation, $\frac{\sigma_S}{\mu_S}$, we can compute this value from the empirical distribution of flow-size and flowlet-size distributions seen in our traces. Fig. 2 compares these values showing that the coefficient of variation for flowlet sizes is always smaller than that for flow size. Thus, overall, deviations from desired splits are an order of magnitude less likely with flowlet-based splitting than with flow-based splitting.

Using the Chebyshev inequality does not usually yield tight bounds. But in our case, Eq. 3 is useful to compare flowlets with flow switching. Also, for a deviation error ϵ, the bound gets tighter as the measurement interval t increases.

Next, we show analytically that the deviation from the desired split is smaller for flowlets than flows.

THEOREM 6.2. *For any sample path of flows, and for which the flows are partitioned into flowlets, the upper bound*

Trace Type	Date	Length (s)	# of Traces	Packets (x 10^6)	# Flows > 5pkts (x 10^3)	Avg. Flow Rate (Kbps)	% Bytes Non-TCP
Peering	Mar'03	720	2	9.15	242.00	6.10	7.97%
LCSout	May'03	3600	336	25.40	102.30	79.16	1.57%
NLANR-1	Mar'04	90	15	7.30	67.81	68.11	13.10%
NLANR-2	Apr'03	90	20	1.69	1.84	185.58	12.10%
Level2-ISP	Apr'03	3600	1	1.48	16.39	5550.00	8.87%
HighCap Peering	Apr'03	284	2	4.06	91.60	13.77	12.50%
LowCap Peering	Oct'02	57600	1	44.87	546.00	34.60	22.40%
LBNL	Sep'05	7200	2	127	810.87	398.79	0.00%
CESCA	Feb'04	300	40	31.8	432.22	46.49	0.75%
CENIC	Mar'05	600	261	19.98	90.61	166.43	1.00%
NLANR-3	Jan'04	300	133	6.48	37.47	199.31	2.11%
Abilene	Jun'04	600	50	61.47	201.17	71.00	15.46%

Table 1: Packet Traces used in evaluation. We used 863 packet traces highly diverse in collection points, flow rates and flow sizes. LCSout, LBNL and CESCA are *access links* at MIT, Lawrence Berkeley Labs and Univ. of Catalunya, Spain. NLANR-1 (610Mbps), NLANR-2 (155Mbps), NLANR-3 (10Gbps), CENIC (10Gbps), Abilene (10Gbps) are *backbone links*. Level2-ISP is an internal link at a small commercial ISP. The rest are *peering links* of different capacities.

in (3) is always smaller for flowlets than flows, i.e.:

$$\frac{\left(\left(\frac{\sigma_S^{flowlet}}{\mu_S^{flowlet}}\right)^2+1\right)}{4\epsilon^2 N(t)^{flowlet}} < \frac{\left(\left(\frac{\sigma_S^{flow}}{\mu_S^{flow}}\right)^2+1\right)}{4\epsilon^2 N(t)^{flow}}, \qquad (4)$$

where for the given sample path, $N(t)$ is the realized number of flows and flowlets, and μ and σ are the realized sample mean and standard variation of flow and flowlet sizes.

Proof The proof is given in Appendix B.

The derivation of the above inequality makes no assumptions about the variability of the flow sizes, but only requires a conservation of mass assumption that the sum of flow sizes equals the sum of flowlet sizes. Thus, inequality 4 holds even for the (non-realistic) hypothetical scenario of constant flow sizes, for which the variance would be 0.

6.2 Analyzing FLARE's Token Counter Algorithm

We now shift attention to the advantages of FLARE's token counting algorithm, described in §5(a). It is known that using feedback about past decisions achieves better accuracy [27]. Here, we have the following conjecture.

CONJECTURE 1. *FLARE's token-counting algorithm has the property that at all times and for all paths, the deviation, measured in bytes, from the ideal split is less than or equal to the maximum size of the splitting unit (packet, flowlets, flows).*

Though we do not have a proof, we have conducted many numerical tests to check the validity of the above conjecture. For example, splitting million random-sized units among ten paths and repeating for thousand different random split functions did not yield a counter-example.

7. EVALUATION ENVIRONMENT

We evaluate FLARE using trace-driven simulations. Our dataset is large and diverse. It contains 863 packet traces collected at tier-1 and regional ISPs, in US and Europe. Table 1 summarizes our dataset.

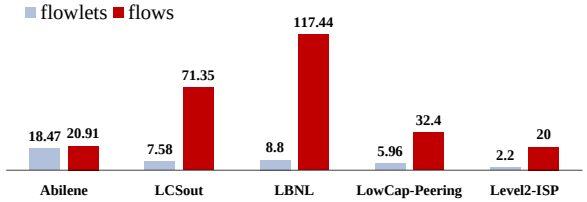


Figure 2: Comparing the Coefficient of Variation (standard deviation/mean) of flowlet size with flow size. Flowlets have a smaller coefficient of variation therefore their deviation from the desired bandwidth shares are also smaller, as shown in Eq. 3.

(a) Compared Splitting Schemes: We compare FLARE against four prior traffic splitting schemes, which are summarized in Table 2.

- *Packet-Based Splitting:* We use Deficit Round Robin to simulate packet-based traffic splitting [27], which is highly accurate [25].
- *Flow-Based Splitting:* We simulate flow-based splitting by assigning each new flow to the path farthest away from its desired traffic share, and retaining the assignment for the duration of the flow.
- *S-HASH Splitting:* We also simulate a static-hash splitting scheme (S-HASH) by hashing the arriving packet's source and destination IP addresses and ports into a large space, then allocating the hash space to the various paths proportionally to their desired traffic shares [25].
- *Bin-Based Splitting:* Finally, we test a bin-based scheme that hashes flows into a small number of bins, typically 16, and assigns bins to paths proportionally to the desired split. This approach is used in Cisco and Juniper routers [11, 17].

(b) Simulation Setup: We assume the router at which the trace is collected to be feeding multiple parallel paths, and splitting the traffic among them according to a desired split vector. We evaluate how well FLARE tracks the desired splits without reordering packets.

Our simulations use three parallel paths for balancing the load. To stress FLARE, we simulate parallel paths with a large difference in their delays. Unless stated differently, the MTBS is 50ms (a value close to coast-to-coast round trip time). Our experiments use both a static vector $\vec{F}_s =$

Technique	Reorder	Overhead	Dynamically Adapt
Packet-Based	Yes	–	Yes
Flow-Based	No	per flow state	mildly
Bin-Based	Yes	# of bins	mildly
Hash-Based	No	–	No
Flowlet-Based	hardly any	5KB Table	Yes

Table 2: The Splitting Techniques Compared in this Paper

$(.3, .3, .4)$, and dynamic vectors

$$\vec{F}_d(t) = .13(1,1,1) + .6(sin^4 x, sin^2 x \cdot cos^2 x, cos^2 x),$$

where $x(t) = \frac{2\pi t}{p}$. In particular, to evaluate responsiveness, we use two dynamic split vectors, $\vec{F}_{d1}$, which reflects changes over long time scales (p=40min), and $\vec{F}_{d2}$, which reflects changes over short time scales (p=4min). The amplitude and period of these vectors are chosen based on [2, 10] to reflect traffic variations reported in the Internet.

(c) Performance Metrics: We use the following metrics.

- *Splitting Error:* An optimal traffic splitting policy ensures that path i receives the desired fraction of the traffic F_i, on any timescale. In practice, the actual fraction of traffic sent on i, F_i', deviates from the desired share. We measure the splitting error as:

$$\begin{aligned} \text{Error} &= \frac{1}{N}\sum_{i=1}^{N}\frac{|F_i - F_i'|}{F_i}, \\ \textit{Accuracy} &= 1 - \textit{Error}. \end{aligned}$$

  Here, N is the number of parallel paths among which the traffic is divided. The graphs report the average error over non-overlapping windows of 300ms. We choose this window because routers usually buffer 300ms worth of traffic. Errors over longer intervals can cause packet drops. Note that splitting error for the dynamic split vectors $\vec{F}_{d1}$ and $\vec{F}_{d2}$ measures responsiveness of the techniques.
- *Reordering Measure:* In our trace-driven simulations, we estimate the disturbance of a certain splitting algorithm as the probability that a packet triggers 3 dup-acks due to the reordering caused by the splitting scheme. While other measures exist [6], we chose this measure because TCP fast-retransmit treats 3-dup-acks as a congestion signal, and reacts to it by entering a recovery state.

(d) Closed-Loop Simulations: Trace-driven simulations capture the characteristics of Internet traffic, but cannot observe how TCP's congestion control interacts with FLARE. Thus, in §9.4, we present the results of closed-loop simulations that use the ns-2 TCP implementation and a FLARE module. We use the packet traces in Table 1 to obtain distributions of flow sizes and flow inter-arrival times, which we use to generate simulated flow arrivals and the corresponding transfer sizes. However, within a flow, the packet arrivals and the congestion windows are dictated by the events that occur during a simulation.

8. EMPIRICAL STUDY OF FLOWLETS

One can argue that switching flowlets from one path to another should not reorder packets. But for this observation to be beneficial, it should be possible to divide a flow into many small flowlets. Even more, since most of the bytes are in long TCP flows [34], for FLARE to achieve good performance the origins of flowlets cannot be limited to short flows, flows that are just starting with a small window of one or two packets, or flows that are suffering timeouts. Unless the long TCP flows arrive in short bursts, flowlet-based splitting will not show a major performance improvement over flow-based splitting.

8.1 Do flowlets really exist?

First, we look at how the number of flowlets increases with the flow size. Fig. 3 plots the average number of flowlets per flow as a function of the flow size. The figure shows data from all traces combined. It shows that large flows do get chopped into many small flowlets. Indeed, even if FLARE sets flowlet timeout to 50ms, (i.e., the paths over which FLARE is balancing the load differ in their delays by as much as 50ms), large flows are still chopped to many small flowlets. Fig. 4 lends further proof to the viability of flowlet switching. It plots the cumulative distribution of traffic bytes vs. flowlet (and flow) sizes over all our traces. Note that more than 60% of the data bytes are contained in small flowlets $< 25KB$, a significant improvement from the about 5% of the bytes that are contained in flows $< 25KB$.

8.2 Where do flowlets come from?

In fact, the main origin of flowlets is the burstiness of TCP at RTT and sub-RTT scales. Prior work has shown that a TCP sender tends to send a whole congestion window in one burst or a few clustered bursts and then wait idle for the rest of its RTT. This behavior is caused by ack compression, slow-start, and other factors [16, 32, 35]. This burstiness enables a FLARE router to consider a long TCP flow as a concatenation of short flowlets separated by idle periods, which is necessary for the success of flowlet-based splitting.

Fig. 5 supports this argument. It is computed using all of our traces for δ=MTBS=50 ms. The figure plots the time between arrivals of two consecutive flowlets from the same flow normalized by the RTT of the flow (RTT is computed using the MYSTERY TCP analyzer [19]). The graph shows that the vast majority of flowlets are separated by less than one RTT, indicating that a flowlet is usually a congestion window or a portion of it.

Though we focus on TCP, evidence from traces shows that flowlets exist even in non-TCP traffic. Perhaps the predominant source of burstiness is interrupt coalescing – modern OS'es handle multiple pending interrupts at a time to reduce the overhead of taking an interrput. This means packets that arrive at a NIC close to each other in time appear to the kernel as if they arrived back to back. Other sources of burstiness include silence suppression (in VoIP over UDP) and user think times (in persistent HTTP).

9. FLARE'S PERFORMANCE

Now that we have established the abundance of flowlets in Internet traffic, we evaluate FLARE's performance.

9.1 FLARE's Accuracy

First, we show that FLARE accurately matches the desired split even when the maximum delay difference between the paths, MTBS, is excessively large. Fig 6 shows the splitting error as a function of flowlet timeout, δ, averaged

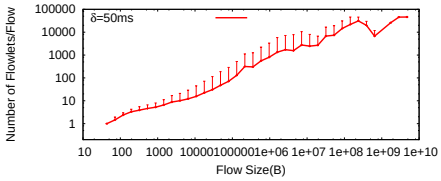


Figure 3: The average number of flowlets in a flow vs. flow size in bytes. Figure shows that large flows that contain most of the bytes also break up into many smaller flowlets.

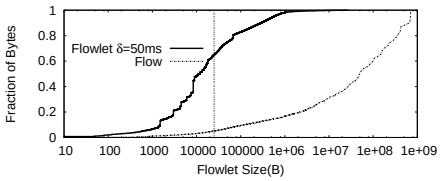


Figure 4: More than 60% of bytes are in flowlets of size smaller than 25KB. This shows that the concept of flowlets shifts most of the bytes to smaller units, which can be independently switched. The solid lines are the fraction of total bytes(CDF) vs. Flowlet size averaged over all traces.

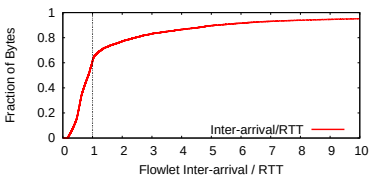


Figure 5: CDF of flowlet interarrival time normalized by the corresponding RTT. About 68% of the bytes are in flowlets that are separated by smaller than one RTT, indicating that most of these flowlets are a whole or a portion of a congestion window.

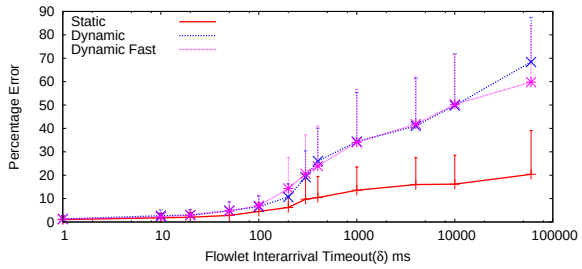


Figure 6: FLARE is accurate. For any MTBS < 100ms, flowlet splitting produces good accuracy. The figure shows the splitting errors as a function of flowlet timeout interval δ for the static, $\vec{F}_s$, the mildly dynamic, $\vec{F}_{d1}$, and the dynamic $\vec{F}_{d2}$ split functions.

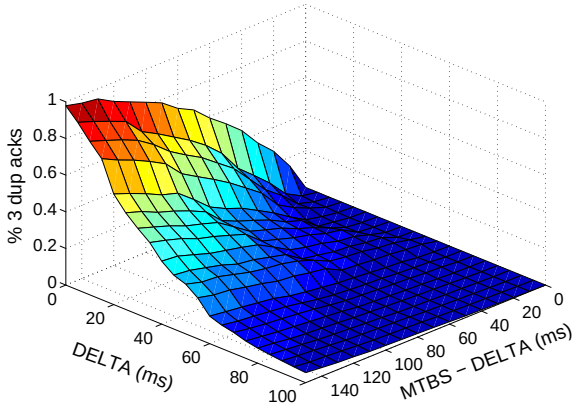


Figure 7: Percentage of packets that lead to three duplicate acks vs. flowlet timeout interval (DELTA δ) and the error in estimating the actual MTBS, $MTBS - \delta$. With packet-based splitting up to 1% of the packets trigger TCP window reduction which is close to the loss rate in the Internet. FLARE with $\delta = 50ms$ causes < .06% re-ordering, even when the actual MTBS is larger than δ by 100ms.

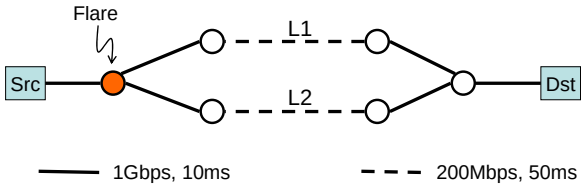


Figure 8: Topology used in the closed-loop experiments.

over all traces. Since FLARE sets δ=MTBS, the figure also presents the error as a function of increased MTBS. The figure shows results for the split vectors: $\vec{F}_s$, $\vec{F}_{d1}$, and $\vec{F}_{d2}$. Note that when $\delta = 0$ we have packet-based splitting, whereas when $\delta \to \infty$ we obtain flow-based splitting. The figure shows that flowlet-based splitting achieves an accuracy comparable to packet-based splitting, as long as δ <100 ms, i.e., the paths' MTBS is less than 100ms. Given typical values for one-way delay in the Internet (e.g., coast-to-coast one-way delay is less than 40ms), a delay difference <100ms should apply to many possible sets of parallel paths. Also note that the errors of flow-based splitting –i.e., $\delta = \infty$– range between 20% and 70% depending on the dynamism of the desired split.

Why Flowlet Splitting is Accurate? First, there are many more flowlets than flows, leading to many opportunities to rebalance an imbalanced load. Table 3 shows that flowlet arrival rates are an order of magnitude higher than flow arrival rates, in our traces. This means that in every second, flowlet-based splitting provides an order of magnitude more opportunities to rebalance an incorrect split than with flow-based splitting. Second, as shown in Fig. 4, most of the bytes are in small flowlets, allowing load rebalancing at a much finer granularity than at the size of a flow.

9.2 Robustness to Errors in MTBS Estimation

FLARE uses periodic pings to estimate the MTBS of the paths across which it balances the load. It sets the flowlet timeout δ to the estimated MTBS. If the MTBS is underestimated, FLARE would operate with a flowlet timeout smaller than the actual MTBS, i.e., $\delta < MTBS$. In this case, there are no guarantees that packet reordering does not occur. This section uses experimental data to show that FLARE is robust to errors in the MTBS estimate.

Fig. 7 shows the probability of mistakenly triggering TCP 3 dup acks, as a function of $MTBS - \delta$ and δ, for the case of 3 paths with a split vector $\vec{F}_s$, and path latencies $(x, x + 0.5\ MTBS, x + MTBS)$. The figure shows that FLARE is tolerant to bad estimates of MTBS, i.e., the probability of 3-dup acks is negligible even for large values of $MTBS - \delta$. In particular, Fig. 7 shows that for any $\delta > 50ms$, the percentage of packets that trigger a TCP window reduction is less than 0.06%, even when the error in estimating the MTBS is as large as 100ms. The impact of such reordering is negligible in comparison with typical drop probabilities in the Internet [23, 3], and thus on average is unlikely to affect TCP's performance.

The combination of the results in this section and in the previous section show that by setting *delta* to

Trace	Arrival Rate (/sec)		Number of Concurrent Items	
	Flows	Flowlets	Flows	Flowlets
LCSout	118.44	1278.84	1477.96 (2030)	29.98 (60)
Peering	611.95	8661.43	8477.33 (8959)	28.08 (56)
NLANR-1	3784.10	35287.04	47883.33 (57860)	240.12 (309)
NLANR-2	111.33	2848.76	1559.33 (1796)	50.66 (71)
Level2-ISP	202.47	295.98	826.76 (931)	1.00 (7)
HighCap-Peering	697.63	7747.06	7080.26 (7683)	54.01 (98)
LowCap-Peering	29.11	345.13	446.69 (769)	5.50 (32)
LBNL	283.18	2177.63	3265.82 (3786)	43.12 (94)
CESCA	4964.36	46407.65	71667.36 (78640)	476.83 (598)
CENIC	305.59	3206.75	3880.80 (4368)	72.49 (112)
NLANR-3	386.00	6838.35	7419.30 (8406)	112.27 (177)
Abilene	2730.70	21807.54	22589.67 (24415)	351.74 (465)

Table 3: Flowlets arrive at a much higher rate than flows; but there are many fewer concurrent flowlets than flows. The values outside parenthesis are averages while the numbers inside are the maximum values.

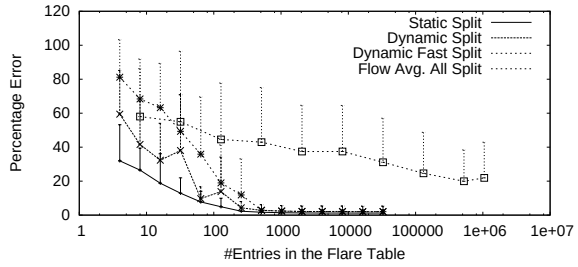


Figure 9: Error as a function of the flowlet table size for the three split vectors, $\vec{F}_s$, $\vec{F}_{d1}$,$\vec{F}_{d2}$. The solid line shows the average error across all traces while the errorbars are one standard deviation long. FLARE achieves low errors without storing much state. A table of 10^3 five-byte entries is enough for the studied traces. The figure also shows the error for flow-switching, averaged over all traces, and the three split vectors. In contrast, error in flow-switching stabilizes only at table size of 10^6.

$max(MTBS\ estimate, 50ms)$, FLARE obtains good accuracy (as shown in Fig. 6) and high robustness to errors in MTBS estimation (as shown in Fig. 7). In §9.5, we support these results further by showing FLARE's accuracy and robustness as functions of the dynamism of the split vector and we compare FLARE with other splitting schemes.

9.3 Negligible Overhead

An important result of this paper is the tiny overhead incurred by flowlet-based splitting. FLARE requires a router to perform a single hash operation per packet and maintain a few KB hash table, which easily fits into the router's cache. We have estimated the required hash table size by plotting the splitting error as a function of the size of the hash table. Fig. 9 shows the error in our traces for both the static split vector $\vec{F}_s$ and the dynamic vectors $\vec{F}_{d1}, \vec{F}_{d2}$. It reveals that the errors converge for a table size as small as 10^3 entries.

Compare the overhead of flowlet-based splitting with flow-based splitting. Flow-based splitting requires per-flow state to maintain the flow-to-path assignment. Fig. 9 shows the average splitting error for flow-based splitting as a function of the hash table size. The table required by flow-based splitting is more than three orders of magnitude larger than flowlet splitting.

Why Does Flowlet Switching Require Only a Small Table? At first, it seems contradictory that we need a smaller table to track flowlets though there are many more flowlets than flows. the large number of flowlets in a trace, FLARE only needs to maintain state for concurrently active flowlets, i.e., flowlets that have packets in flight, in the network. Table 3 shows that the average number of concurrent flowlets is many orders of magnitude smaller than the number of concurrent flows. Indeed the maximum number of concurrent flowlets in our traces never exceeds a few hundreds. To track these flowlets without collision, the router needs a hash table of about a thousand entries, which is relatively cheap.

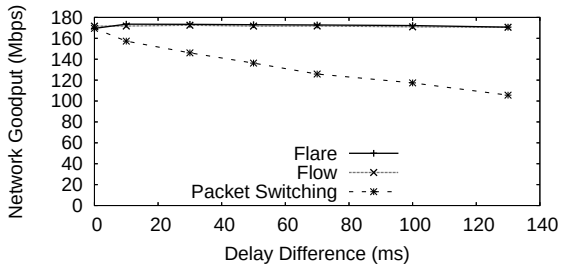


Figure 10: Closed-Loop Simulation showing that packet switching hurts end2end goodput considerably. Total goodput of the network vs. delay difference between the set of parallel paths. As delay difference increases, packet switching is more likely to re-order packets huring the goodput of TCP flows.

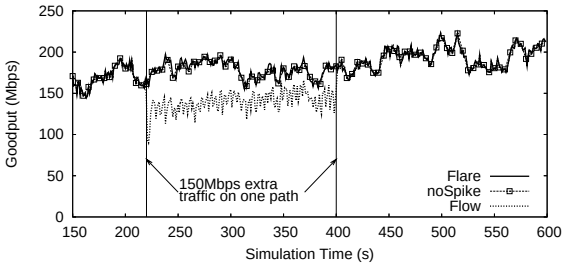
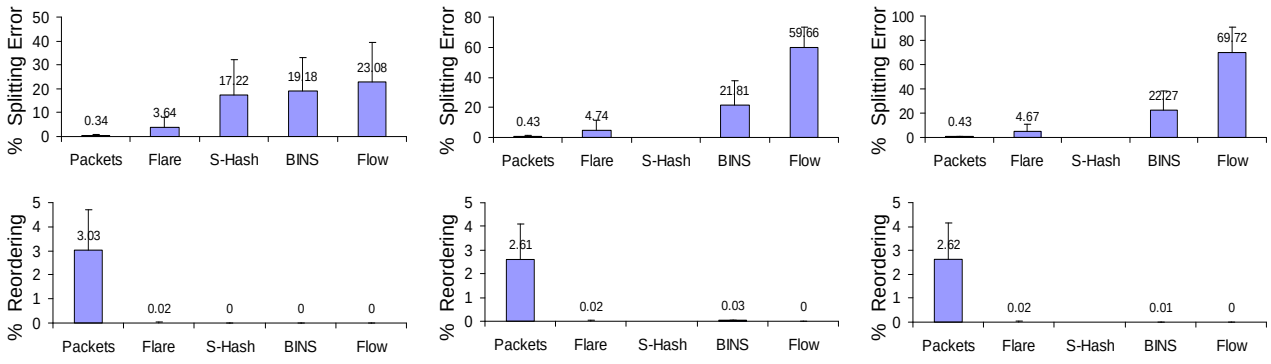


Figure 11: Closed-Loop Simulation shows that the inaccuracy of flow-based splitting hurts end2end goodput. Between $[220, 400]s$, one of the paths in Fig. 8 sees an extra 150Mbps of cross traffic. Traffic is split 20%-80% during this period to account for the change in available bandwidth (50Mbps on one path and 200Mbps on the other). Flow-splitting results in less goodput during this period as it is unable to move ongoing flows away from the congested path.

9.4 Closed-Loop Simulations of FLARE

Trace-driven simulations capture the characteristics of Internet traffic, but cannot observe how TCP's congestion control interacts with FLARE. To investigate that issue, we implement a closed-loop simulator in *ns-2*. We use a simple topology with two parallel paths shown in Fig. 8. the traffic split ratio $\vec{F}_s = (0.5, 0.5)$ matches the bottleneck capacities: $L1 = L2 = 200Mb/s$. A splitting agent (e.g., FLARE) splits traffic across these parallel paths. Our simulations have bandwidth delay product buffers, DropTail queues, 1500 byte packets, and traffic on the reverse paths. We play the traces in §7 to generate the flow arrivals and the corresponding transfer sizes. Flows that stall for more than one minute are terminated to prevent the number of flows in the simulation from increasing indefinitely. Packet arrivals



(a) CONSTANT Splits ($\vec{F}_s$) (b) Mildly Dynamic Splits ($\vec{F}_{d1}$) (c) Dynamic Splits ($\vec{F}_{d2}$)

Figure 12: Comparison between various splitting schemes. FLARE is significantly more accurate than flow-based, S-HASH, and BIN-based splitting schemes and its robustness to reordering is an order of magnitude higher than packet-based splitting. We show the mean (solid bars) and standard deviation (thin lines) of the splitting error and percentage of 3-dupacks averaged across traces for static, mildly dynamic, and dynamic split vectors. The experiment uses δ=50 ms and MTBS=70 ms. FLARE's reorderings arise from this assumed estimation error. When δ=MTBS, FLARE shows no reordering. **Note S-HASH cannot be run with dynamic splits.**

and the congestion windows are dictated by the events that occur during a simulation.

Our closed-loop simulations, detailed below, show that:

- FLARE interacts benignly with TCP timers and congestion control. In particular, splitting a TCP flow across multiple paths using flowlet switching has no negative impact on TCP throughput.
- End-to-end goodput improves with FLARE. Packet-based splitting reduces the end-to-end goodput when the parallel paths differ in latency, whereas flow-based splitting hurts end-to-end goodput when the traffic splits change dynamically.

While, the results in the previous sections focus on splitting errors and the probability of 3 dup-acks, the results below focus on end-to-end performance, particularly network goodput. Fig. 10 evaluates the impact of reordering on goodput as the delay difference between the two paths increases. The figure reveals two important results. First, reordering caused by packet-based splitting significantly hurts goodput. In fact, high-rate flows suffer the most because their larger congestion windows make it more likely that reordering will cause three dupacks. Second, FLARE can split the same TCP flow down paths whose one-way latencies are different by as much as 100ms without adversely interacting with TCP timers or hurting TCP throughput. This is because flows tend to use both paths and hence, the averaging employed by TCP timers ends up with a weighted average of the two latencies. Further, the TCP RTO estimator is highly conservative–it uses four times the variance in RTT [24].

Fig. 11 shows that the poor accuracy of flow splitting hurts network goodput. Between $t = [220, 400]s$, one of the paths sees an extra 150Mbps load. During this period, traffic that was split evenly between the two paths is now split at 20%-80% because the loaded path only has 50Mbps of available capacity in contrast to 200Mbps on the unloaded path. Flow-splitting cannot move ongoing flows away from the congested path. This congests the overloaded path, results in a high drop rate, and reduces the overall goodput. FLARE, on the other hand, reacts quickly and ensures that both ongoing and new flows get to spend some time on each of the two paths avoiding any flow from being starved. It is worth noting here that Flare succeeds in splitting a TCP flow down multiple paths that have different available bandwidth and is not affected by transient congestion.

9.5 Comparison with Other Splitting Schemes

We compare FLARE with other splitting schemes vis-a-vis splitting errors and TCP disturbance. The experiments are for δ=50ms and MTBS=70ms, which accounts for potential errors in the estimation of MTBS.

Fig. 12 shows the results for static, mild dynamic, and dynamic split vectors. The figure shows that FLARE is a good tradeoff between accuracy, responsiveness, and robustness to reordering. Its errors are significantly lower than flow-based, S-HASH, and BIN-based splitting schemes, and its tendency to trigger TCP congestion window reduction is negligible.

FLARE's advantages are most prominent when split vectors are dynamic. In this case, FLARE is almost as responsive as packet-based splitting. In contrast, flow-based splitting overshoots the desired splits by 60%, on average. Packet-based splitting triggers 3 dup ack events at a rate comparable to or higher than the loss rate in the Internet [23, 5]. The S-HASH scheme, has a reasonable splitting accuracy for a static split vector, but does not react to dynamic split vectors. The BIN-based splitting scheme, cannot split traffic at granularity smaller than $\frac{1}{\#\text{ of bins}}$ and allows packets to be re-ordered when a bin is moved from one path to another.

10. CONCLUDING STATEMENTS

We have introduced the concept of flowlet-switching and developed an algorithm which utilizes flowlets in traffic splitting. Our work reveals several interesting conclusions. First, highly accurate traffic splitting can be implemented with little to no impact on TCP packet reordering and with negligible state overhead. Next, flowlets can be used to make load balancing more responsive, and thus help enable a new generation of realtime adaptive traffic engineering. Finally, the existence and usefulness of flowlets show that TCP burstiness is not necessarily a bad thing, and can in fact be used advantageously.

11. REFERENCES

[1] A. Aggarwal, S. Savage, and T. Anderson. Understanding the performance of TCP pacing. In *INFOCOM*, 2000.

[2] A. Akella, S. Seshan, and A. Shaikh. Multihoming Performance Benefits: An Experimental Evaluation of Practical Enterprise Strategies. In *USENIX Tech. Conf.*, 2004.

[3] M. Allman, W. Eddy, and S. Ostermann. Estimating loss rates with tcp. *ACM Performance Evaluation Review*, 2003.

[4] S. Amstutz. Burst switching—an update. In *IEEE Commun. Mag.*, 1986.

[5] D. Andersen, A. Snoeren, and H. Balakrishnan. Best-path vs. multi-path overlay routing. In *ACM IMC*, 2003.

[6] J. C. R. Bennett, C. Partridge, and N. Shectman. Packet Reordering is not Pathological Network Behavior. In *Transactions on Networking*, 1999.

[7] E. Blanton and M. Allman. On making tcp more robust to packet reordering. In *ACM Computer Communication Review*, 2002.

[8] J. E. Burns, T. J. Ott, A. E. Krzesinski, and K. E. Muller. Path Selection and Bandwidth Allocation in MPLS Networks. *Perform. Eval*, 2003.

[9] Z. Cao, Z. Wang, and E. W. Zegura. Performance of hashing-based schemes for internet load balancing. In *IEEE INFOCOM*, 2000.

[10] C. N. Chuah and C. Diot. A tier-1 isp prespective: Design principles & observations of routing behavior. In *PAM*, 2002.

[11] Cisco express forwarding (cef). Cisco white paper, Cisco Systems., July 2002.

[12] A. Elwalid, C. Jin, S. H. Low, and I. Widjaja. MATE: MPLS Adaptive Traffic Engineering. In *INFOCOM*, 2001.

[13] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights in a Changing World. In *INFOCOM*, 2000.

[14] B. Fortz and M. Thorup. Optimizing OSPF Weights in a Changing World. In *IEEE JSAC*, 2002.

[15] R. G. Gallager. In *Discrete Stochatic Processes*, 1996.

[16] H. Jiang and C. Dovrolis. The origin of tcp traffic burstiness in short time scales. Technical report, Georgia Tech., 2004.

[17] Junos 6.3 internet software routing protocols configuration guide. www.juniper.net/techpubs/software/junos/junos63/swconfig63-routing/html/.

[18] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the Tightrope: Responsive yet Stable Traffic Engineering. In *SIGCOMM*, 2005.

[19] S. Katti, C. Blake, D. Katabi, E. Kohler, and J. Strauss. M&M: Passive measurement tools for internet modeling. In *ACM IMC*, 2004.

[20] M. Laor and L. Gendel. The effect of packet reordering in a backbone link on application throughput. *IEEE Network*, 2002.

[21] R. Ludwig and R. Katz. The eifel algorithm: Making TCP robust against spurious retransmissions. In *ACM CCR*, 2000.

[22] K. Papagiannaki, N. Taft, and C. Diot. Impact of flow dynamics on traffic engineering design principles. In *INFOCOM*, Hong Kong, March 2004.

[23] V. Paxson. End-to-end internet packet dynamics. *ACM TON*, 1999.

[24] V. Paxson and M. Allman. Computing tcp's retransmission timer, 2000. IETF RFC 2988.

[25] S. Rost and H. Balakrishnan. Rate-aware splitting of aggregate traffic. Technical report, MIT, 2003.

[26] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. In *IMW*, 2002.

[27] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. In *SIGCOMM*, 1995.

[28] C. Villamizar. MPLS Optimized Multipath (MPLS-OMP), 1999. Internet Draft.

[29] C. Villamizar. Ospf Optimized Multipath (OSPF-OMP), 1999. Internet Draft.

[30] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. Cope: Traffic engineering in dynamic networks. In *ACM SIGCOMM*, 2006.

[31] Y. Wang and Z. Wang. Explicit routing algorithms for internet traffic engineering. In *IEEE ICCCN*, 1999.

[32] L. Zhang, S. Shenker, and D. D. Clark. Observations on the dynamics of a congestion control algorithm. In *SIGCOMM*, 1991.

[33] M. Zhang et al. RR-TCP: A reordering-robust tcp with dsack. In *IEEE ICNP*, 2003.

[34] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the characteristics and origins of internet flow rates. In *SIGCOMM*, 2002.

[35] Z.-L. Zhang, V. Ribeiro, S. Moon, and C. Diot. Small-time scaling behaviors of internet backbone traffic. In *INFOCOM*, 2003.

Appendix A: How far can probabilistic assignment be from the desired split ratio?

As a conceptual model for the fractional splitting of traffic, consider placing balls sequentially in one of m bins. The bins represent the set of parallel paths, whereas the balls refer to the splitting unit which could be a packet, a flow, or a flowlet. The ball size, S_j refers to the size of the splitting units, and hence follows a particular distribution: packet size distribution, flow size distribution, or flowlet size distribution. For simplicity, we assume the sizes of the various balls are independent and identically distributed random variables. Let $N(t)$ denote a random variable for the number of balls to arrive over a time interval (0, t]. For the following it is not needed to specify the nature of the arrival process of balls, though we do assume that $N(t)$ is independent of the ball sizes, S_j.[2] The objective is to assign the $N(t)$ balls to the bins such that the total load is split between the bins according to some given split ratios $\vec{f} = (f_1, \ldots, f_m)$, $\sum_1^m f_i = 1$.

Assume a simple, probabilistic scheme in which each ball is tossed into bin i with probability f_i, equal to the desired split ratio. Focus on an arbitrary bin i, and let I_j^i be an indicator variable that denotes the event that ball j went into bin i. Then, $I_j^i = 1$ with probability f_i and is 0 otherwise. Further, I_j^i is independent of the indicator variables of other balls. Note,

$$E[I_j^i - f_i] = 0, \quad \text{and} \quad E[(I_j^i - f_i)^2] = f_i(1 - f_i). \tag{5}$$

Suppose we continue splitting over a time interval (0, t] during which $N(t)$ balls arrive. The number of bytes sent down tunnel i during this interval is:

$$B_i(t) = \sum_{j=1}^{N(t)} I_j^i S_j,$$

In contrast, the desired split is $f_i \sum_{j=1}^{N(t)} S_j$ and the deviation from the desired split, normalized by the expected total traffic volume is:

$$D_i(t) = \frac{\sum_{j=1}^{N(t)} (I_j^i - f_i) S_j}{E\left[\sum_{j=1}^{N(t)} S_j\right]}.$$

Applying the strong law of large numbers to the right hand side above and noting that the expected value of $I_j^i - f_i$ is zero, (and given that the number of balls to arrive, $N(t)$ approaches ∞ as $t \to \infty$ and the expected ball size, $E[S_j]$, is finite), then the deviation $D_i(t)$ approaches 0 with probability one, as $t \to \infty$.

[2] Alternatively, we could have defined N in terms of bytes and not time; in particular we could define $N(b)$ to be the ball number for which the sum of ball sizes first equals or exceeds b bytes. The analysis would follow the same logic and yield the same upper bound, Eq. 13.

Thus, if hypothetically one had an infinite number of balls, then this simple probabilistic assignment of balls to bins would achieve the desired fractional assignment.

Of more interest, we wish to bound the probability that the normalized deviation for an arbitrary bin is larger than ϵ at an arbitrary, finite time t. For this, we use the bound due to Chebyshev [15]:

$$P(|D_i(t) - E[D_i(t)]| > \epsilon) < \frac{\sigma^2_{D_i}}{\epsilon^2}, \quad (6)$$

where $\sigma^2_{D_i}$ is the variance of the deviation $D_i(t)$.

Note that $E(D_i|N(t) = n, \vec{S} = \vec{s})$, for any given choice of number of balls and ball sizes, $\vec{s}$, is

$$= \frac{\sum_{j=1}^{n} E\left[(I_j^i - f_i)s_j\right]}{E\left[\sum_{j=1}^{N(t)} S_j\right]} = \frac{\sum_{j=1}^{n} s_j}{E\left[\sum_{j=1}^{N(t)} S_j\right]} \cdot E[I_j^i - f_i] = 0. \quad (7)$$

This is due to linearity of expectation and from Eq. 5. This leads to,

$$E(D_i) = E_{N(t),\ \vec{S}}\left[E(D_i|N(t) = n, \vec{S} = \vec{s})\right] = E_{N(t),\ \vec{S}}[0] = 0. \quad (8)$$

Similarly $E(D_i \cdot D_i|N(t) = n, \vec{S} = \vec{s})$

$$= \frac{E\left[\sum_j s_j^2 \cdot (I_j^i - f_i)^2 + \sum_{j!=k} s_j s_k \cdot (I_j^i - f_i) \cdot (I_k^i - f_i)\right]}{(E[\sum_{j=1}^{N(t)} S_j])^2},$$

$$= \frac{\sum_j s_j^2 \cdot f_i(1 - f_i)}{(E[\sum_{j=1}^{N(t)} S_j])^2}. \quad (9)$$

The first part is due to expanding the product of $D_i \cdot D_i$ into n square terms and $n^2 - n$ product terms. The second part is due to Eq. 5. The product terms cancel out since I_j^i and I_k^i are independent. This immediately leads to $E(D_i \cdot D_i)$

$$= f_i(1 - f_i) \cdot \frac{E\left[\sum_{j=1}^{N(t)} S_j^2\right]}{\left(E[\sum_{j=1}^{N(t)} S_j]\right)^2}$$

$$= \frac{f_i(1 - f_i) \cdot E[N(t)] \cdot E[S_j^2]}{(E[N(t)] \cdot E[S_j])^2}$$

$$\leq \frac{E[S_j^2]}{4E[N(t)] \cdot (E[S_j])^2}. \quad (10)$$

For the above, we use the independence of $N(t)$ and S_j and that since $f_i \in [0, 1]$, $f_i(1 - f_i) \leq \frac{1}{4}$.

From Eqs. 8, 10, the variance

$$\sigma^2_{D_i} = E[D_i.D_i] - E^2[D_i] \leq \frac{E[S_j^2]}{4E[N(t)] \cdot (E[S_j])^2}. \quad (11)$$

Plugging Eqs. 8, 11 into Eq. 6 yields

$$P(|D_i(t)| > \epsilon) < \frac{E[S_j^2]}{4\epsilon^2 E[N(t)]E^2[S_j]} \quad (12)$$

$$= \frac{1}{4\epsilon^2 E[N(t)]}\left((\frac{\sigma_S}{\mu_S})^2 + 1\right). \quad (13)$$

Eq. 13 is obtained from 12 by subtracting and adding $E^2[S_j]$ to the numerator, and where σ_S and μ_S denote respectively the standard deviation and mean of the ball size. The ratio σ_S/μ_S is known as the coefficient of variation and is a common, normalized measure for the variability of a random variable.

Appendix B: Comparing bounds on deviation for Flowlets and Flows

We prove the inequality 4 in Section 6, showing that the bound on deviation is tighter for flowlets than for flows based on a sample-path argument and the "conservation of mass" assumption that for any sample path:

$$\textit{sum of flow sizes} = \textit{sum of flowlet sizes} \quad (14)$$

and to exclude the trivial case where each flow is "partitioned" into just one flowlet, we assume that at least one flow is indeed partitioned into at least two flowlets.

Let sample average number of flowlets in a flow, denoted $\bar{N}$ be:

$$\bar{N} = \frac{\textit{number of flowlets}}{\textit{number of flows}}. \quad (15)$$

From Eq. 14, 15, the sample means of flow and flowlet sizes, denoted respectively $\mathrm{m}[\mathrm{S}^{\mathrm{flow}}]$ and $\mathrm{m}[\mathrm{S}^{\mathrm{flowlet}}]$ are related as follows:

$$\mathrm{m}[\mathrm{S}^{\mathrm{flow}}] = \frac{\textit{sum of flow sizes}}{\textit{number of flows}} = \frac{\textit{sum of flowlet sizes}}{\textit{number of flowlets}/\bar{N}}$$

$$= \bar{N} \cdot \frac{\textit{sum of flowlet sizes}}{\textit{number of flowlets}} = \bar{N} \cdot \mathrm{m}[\mathrm{S}^{\mathrm{flowlet}}] \quad (16)$$

For the second moments, consider first an arbitrary flow size, S_j^{flow}, which has been partitioned into N_j flowlets and where $S_{kj}^{flowlet}$ is the size of the k^{th} flowlet in the flow j. The square of the flow size is:

$$\left(S_j^{flow}\right)^2 = \left(\sum_{k=1}^{N_j} S_{kj}^{flowlet}\right)^2 > \sum_{k=1}^{N_j}\left(S_{kj}^{flowlet}\right)^2. \quad (17)$$

The last inequality is due to the fact that the square of the sum of positive values is greater than the sum of the squares of these values and is the key for the next step.

Let us denote the sample second moments respectively as $\mathrm{m}\left[(\mathrm{S}^{\mathrm{flow}})^2\right]$ and $\mathrm{m}\left[(\mathrm{S}^{\mathrm{flowlet}})^2\right]$, and the number of flows and flowlets respectively as N^{flow} and $N^{flowlet}$. Summing up both sides of Eq. 17 over all flows yields:

$$\mathrm{m}[(\mathrm{S}^{\mathrm{flow}})^2] \equiv \frac{1}{N^{flow}}\sum_{j=1}^{N^{flow}}\left(S_j^{flow}\right)^2 > \frac{1}{N^{flow}}\sum_{j=1}^{N^{flow}}\sum_{k=1}^{N_j}\left(S_{kj}^{flowlet}\right)^2$$

$$= \frac{\bar{N}}{N^{flowlet}}\sum_{j=1}^{N^{flow}}\sum_{k=1}^{N_j}\left(S_{kj}^{flowlet}\right)^2 = \bar{N} \cdot \mathrm{m}[(\mathrm{S}^{\mathrm{flowlet}})^2] \quad (18)$$

where the double sum on the right hand side is the sum over all flowlets of the square of the flowlet sizes.

Dividing both sides of Eq. 18 by the square of the sample mean flow size, Eq. 16, we obtain:

$$\frac{\mathrm{m}[(\mathrm{S}^{\mathrm{flow}})^2]}{\left(\mathrm{m}[\mathrm{S}^{\mathrm{flow}}]\right)^2} > \frac{1}{\bar{N}} \cdot \frac{\mathrm{m}[(\mathrm{S}^{\mathrm{flowlet}})^2]}{\left(\mathrm{m}[\mathrm{S}^{\mathrm{flowlet}}]\right)^2} \quad (19)$$

The format of the inequality 4 is obtained easily from Eq. 19 by noting that the second moment is the sum of squared mean and variance, dividing both sides by $4\epsilon^2 N(t)^{flow}$ and using Eq. 15.