Balancing by PREFLEX: Congestion Aware Traffic Engineering

João Taveira Araújo, Richard Clegg, Imad Grandi, Miguel Rio, and George Pavlou

Network & Services Research Laboratory
Department of Electronic & Electrical Engineering
University College London, UK
{j.araujo, r.clegg, i.grandi, m.rio, g.pavlou}@ee.ucl.ac.uk

Abstract. There has long been a need for a robust and reliable system which distributes traffic across multiple paths. In particular such a system must rarely reorder packets, must not require per-flow state, must cope with different paths having different bandwidths and must be self-tuning in a variety of network contexts. PREFLEX, proposed herein, uses estimates of loss rate to balance congestion. This paper describes a method of automatically adjusting how PREFLEX will split traffic in order to balance loss across multiple paths in a variety of network conditions.
Equations are derived for the automatic tuning of the time scale and traffic split at a decision point. The algorithms described allow the load balancer to self-tune to the network conditions. The calculations are simple and do not place a large burden on a router which would implement the algorithm. The algorithm is evaluated by simulation using ns-3 and is shown to perform well in a variety of circumstances. The resulting adaptive, end-to-end traffic balancing architecture provides the necessary framework to meet the increasing demands of users while simultaneously offering edge networks more fine-grained control at far shorter timescales.

Key words: Traffic engineering, congestion control

1 Introduction

Traffic engineering, whether applied intradomain [1, 2] or interdomain [3–5], focuses on distributing traffic across multiple paths whilst reducing an explicit cost, typically by minimizing the maximum link load observed over a period of time. Most research within traffic engineering however has been limited to optimizing traffic distribution from the vantage point of a single domain [6], failing to take into consideration the wider impact such actions may have on end-to-end performance. This myopic view of traffic has become a severe limitation as end-hosts become increasingly able to pool multiple paths according to their own preferences, optimizing for throughput, latency or reliability. While such trends are already noticeable due to peer-to-peer applications such as Bittorrent, it is expected that resource pooling will further tip toward the end-host with the deployment of multipath transport protocols such as MPTCP [7].

As a consequence of traffic patterns becoming more dynamic, traffic engineering has been forced to remain static. Since both congestion control and traffic engineering are intrinsically tied but unaware of each other by nature, there is a justified concern that updating routing too often may lead to oscillations in demand as hosts shift traffic to paths with better end-to-end performance. Coupled with an aversion to costly upgrades, this conservative approach to traffic management has meant that despite proposals for dynamic, online traffic engineering [8, 9], such solutions have remained both untested and undeployed.

While interest in traffic engineering has steadily waned within the research community due to these perceived limitations, the demand for fine grained traffic management tools remains high, as demonstrated by the continued proliferation of tools such as deep-packet inspection in order to shape traffic explicitly. Such mechanisms illustrate the continued tussle to circumvent the opaqueness inherent to the Internet's layered architecture. Networks frequently attempt to act on information contained within the transport or application layer in order to override resource allocation as performed by TCP. Conversely, the transport layer often makes incorrect assumptions due to the limited information contained within the network layer. With no explicit information on the path taken by each packet, TCP assumes a single path is maintained for the entire flow, and as such has grown to expect in-order delivery from the IP layer [10].

PREFLEX [11] is an architecture which addresses this information asymmetry by providing hosts with explicit knowledge of paths and networks with insight into the loss rate experienced by TCP. PREFLEX provides the means by which edge networks can assign flows to paths, but has not yet defined a method for such an assignment. In this paper we formulate an autonomous tuning mechanism for balancing traffic according to expected congestion rather than load.

2 PREFLEX overview

PREFLEX (Path RE-Feedback with Loss EXposure) [11] is an architecture which redraws the boundaries between transport and network layers in an attempt to make both aware of each other's actions. PREFLEX relies on three components:

1. Hosts provide the network with an estimate of loss using LEX (Loss EXposure).
2. Edge networks provide hosts with information on which outgoing path to use through PREF (Path RE-Feedback).
3. A mechanism which assigns "flowlets" [12] to paths in order to balance loss.

The latter is the focus of this paper. In the following sections we will briefly review the first two components.

2.1 Loss Exposure

Loss exposure is a mechanism by which hosts signal transport semantics at the network layer by marking the IP header, in a similar fashion to re-ECN [13]. If a host does not support LEX, all traffic is unmarked and will appear to the network as legacy traffic. For LEX-capable traffic, there are three possible codepoints:

Codepoint	Meaning
Not-LECT	Not Loss Exposure Capable Transport
LECT	Loss Exposure Capable Transport
LEx	Loss Experienced
FNE	Feedback Not Established

Table 1. LEX markings.

The FNE codepoint is set by the host when feedback on the path has not yet been established, and therefore there is no congestion control loop. This applies to the first packet of a flow, such as TCP SYN packets, or the first packet after a long idle time, such as keep-alives. An important distinction is that FNE marks the beginning of a “flowlet” [12], rather than a flow. We define flowlets as a burst of packets which the sender does not wish to be reordered. As such, it is the end-hosts decision on how and when to divide traffic into flowlets. While a flow is a transport association between two endpoints, a flowlet is a sequence of packets which is expected to follow the same path. As an example, an IMAP connection may remain open several hours, but is composed of several distinct flowlets each time a client requests data from the server. Operating on flowlets rather than flows allows networks to balance traffic at a finer granularity and in a transport agnostic manner.

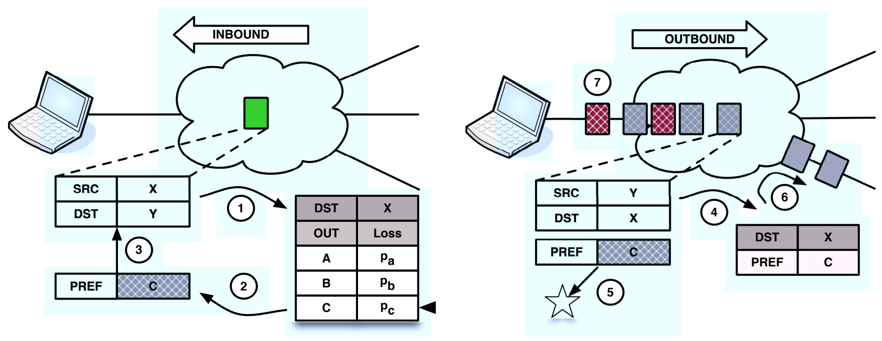
The remaining two codepoints, LECT and LEx in table 1, are used to signal path loss. By default, traffic is marked as being Loss Exposure capable, and for every loss experienced by a host, the ensuing packet is marked with the Loss Experienced codepoint. In reliable transport protocols such as TCP this corresponds to marking every retransmitted packet accordingly, but can also be applied to non-reliable transport protocols with some form of feedback on loss, such as DCCP [14].

2.2 Path Re-Feedback

With LEX, hosts provide information on both path loss and flowlet start at the network layer. We now focus on how edge networks specifically, which are naturally multi-homed, can reflect path diversity back to hosts.

In PREF (Path RE-Feedback) the network selects a preferred outgoing path for each incoming FNE packet (fig. 1(a)). Upon receiving the first packet of a flowlet, the host performs a reverse lookup on the source address (step 1) and selects a path according to the perceived performance (2). The router then associates the chosen path identifier to the packet and forwards the packet toward the host (3).

The treatment of outbound traffic by PREFLEX is illustrated in figure 1(b). The host, having received indication of the preferred path, tags all traffic deemed sensitive to reordering with the given path identifier. As the PREFLEX aware router receives this marked traffic it updates statistics associated to path, aggregating loss for each destination prefix (4). It then discards the path identifier (5) and forwards traffic along the appropriate path (6).



(a) Inbound FNE packet (b) Outbound traffic

Fig. 1. PREFLEX architecture.

The resulting architecture provides many benefits. The balancing is both transport agnostic and allows flow state to be kept to a minimum, requiring policing at the edges only if congestion accountability is required. Additionally, path selection is receiver driven, aligning the stakeholder who can decide when to issue FNE packets with the stakeholder who benefits the most. The key to path selection is the information provided by the sender (point 7), which allows the network to calculate the percentage of path loss. We assume for the entirety of this paper that each flow follows a single path and that a host does not override the path selected by its network, although neither is strictly necessary. The implications of both assumptions being broken and the incentives involved are briefly reviewed in [11].

3 Balancing congestion

While PREFLEX provides an architecture within which transport and network layers exchange relevant information, it does not yet define how networks select outgoing paths. In this section we present a solution for balancing traffic according to expected congestion which works in a wide variety of network conditions.

3.1 Model

Consider LEX-capable traffic, that is explicitly marked as either being retransmitted or non-retransmitted, from a single origin prefix to a single destination prefix, with a number of possible paths and within a single time period. Let N be the number of paths (numbered $1, \ldots, N$). Let T_i be the number of bytes sent down path i for the previous time period. Let R_i be the number of bytes marked as retransmissions down path i for the previous time period. Let $R = \sum_i R_i$ and $T = \sum_i T_i$. While R_i does not strictly represent the number of lost bytes, the ratio of R_i/T_i should be a good approximation of the loss rate within period i. A starting assumption is that it is desirable to equalise the proportion of lost bytes on all interfaces – that is make R_i/T_i equal for all i – and by doing so loss is balanced.

The loss rate $\rho_i = R_i/T_i$ is an unknown function of T_i and B_i the bandwidth of the link. It is, however, likely that the loss rate is increasing (or at least non-decreasing) with T_i. Similarly, the loss rate is decreasing (or at least non-increasing) with the unknown B_i. Assume then that whatever the true function, for a small region around the current values of T_i and B_i then it is locally linear $\rho_i = k_i T_i/B_i$ where k_i is an unknown constant. Substituting gives

$$B_i/k_i = T_i^2/R_i. \tag{1}$$

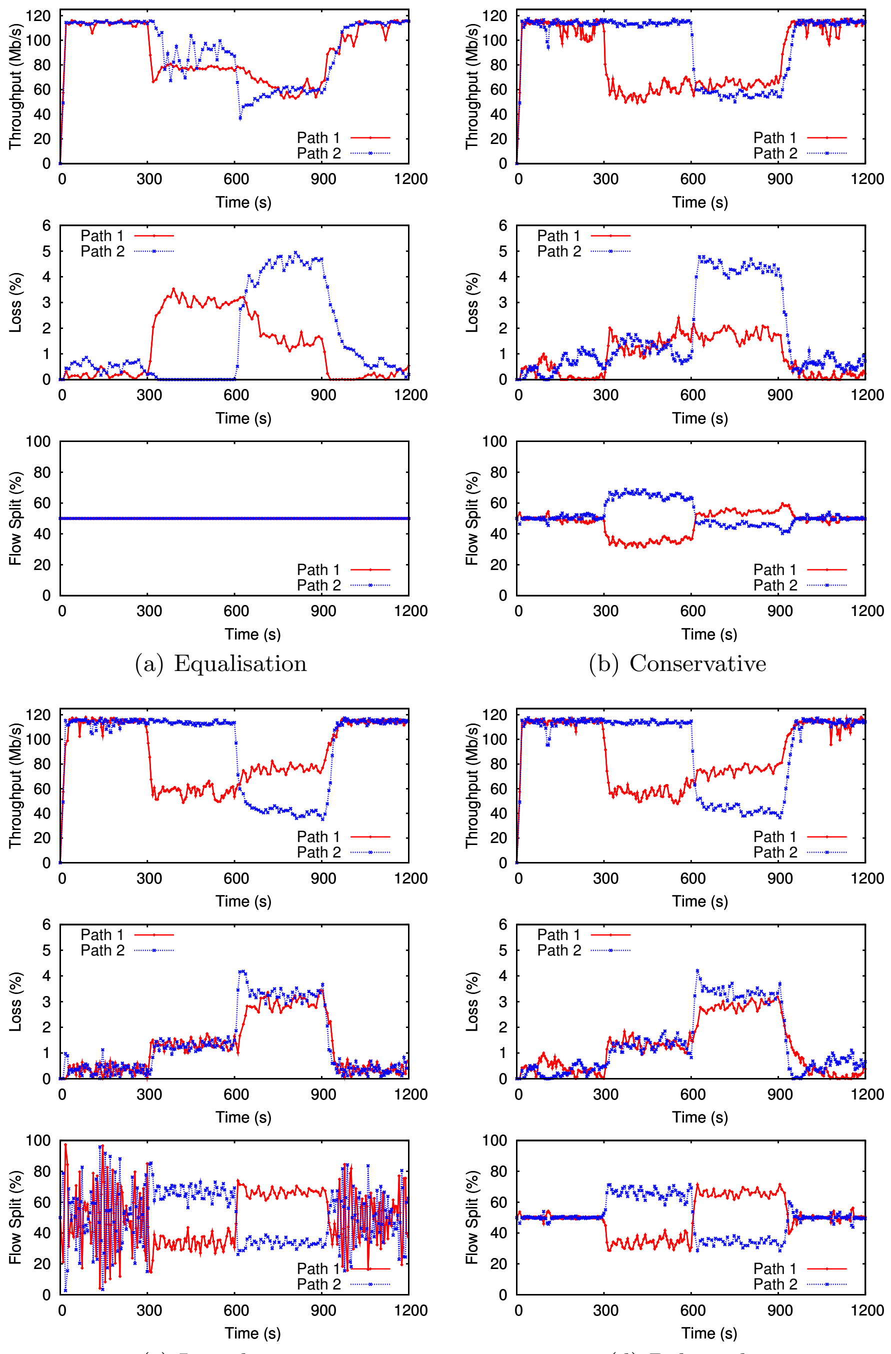
Now consider the next time period. Use the dash notation for the same quantities in the next time period. In typical time periods $\mathrm{E}\left[T'\right] = \mathrm{E}\left[T\right]$ (since, for example, if the next time period on average had more traffic, the overall amount of traffic would be growing – while this is true in the year-on-year setting, this effect is negligible in the time scale discussed here). Now $\rho_i' = R_i'/T_i' = kT_i'/B_i'$. Choose the T_i' to make all $R_i'/T_i' = C$ (hence all equal) where C is some unknown constant. Therefore $T_i' = CB_i'/k_i'$ but if this is still near our locally linear region then $B_i'/k_i' \simeq B_i/k_i$ and B_i/k_i is determined by (1) hence the predicted distribution is $T_i' = CT_i^2/R_i$. Now it is necessary to calculate C by summing over i. $T = C\sum_i T_i^2/R_i$ and hence $C = T/\sum_i T_i^2/R_i$. This gives the final answer:

$$T_i' = \frac{TT_i^2}{R_i \sum_j (T_j^2/R_j)}. \tag{2}$$

There is, of course, a problem when $R_i = 0$. Because of the assumption that the traffic is assigned in inverse proportion to loss rate then a branch with no loss would naturally have all the traffic assigned to it. To correct this, we increment R_i for each path by one maximum segment size (MSS). This problem will be further discussed in subsequent sections.

3.2 Understanding the design space

Because only local linearity is assumed then large adjustments of the traffic split are likely to cause problems. It is also useful to send a small amount of probe traffic down each route even if the loss rate is high [15]. In the absence of other information (for example when there is no loss) it is useful to assign traffic according to the current traffic throughput split. Therefore there are three tendencies to be accounted for, the loss-equalisation tendency from (2), the conservative tendency to keep the traffic split the same as the throughput split in the previous period and the equalisation tendency to keep the traffic equally split on all interfaces. Call the traffic split assigned to path i by each of these schemes



(a) Equalisation (b) Conservative

(c) Loss-driven (d) Balanced

Fig. 2. Simulation using PREFLEX to balance traffic over two paths. Each simulation demonstrates the dynamics of a single mode, except for (d) which balances between conservative and loss-driven components.

$T(E)_i$, $T(C)_i$ and $T(L)_i$ where E, C and L stand for "equal", "conservative" and "loss-driven".

Then our final distribution of traffic across all links is:

$$T'_i = \beta_E T'(E)_i + \beta_C T'(C)_i + \beta_L T'(L)_i,$$

where the $\beta_\bullet$ are user set parameters in $(0, 1)$ such that $\beta_E + \beta_C + \beta_L = 1$. Now $T'(C)_i = T_i$ and $T'(E)_i = T/N$ where N is the number of interfaces. This gives a first equation for the desired flow split f'_i, which represents the probability with which path i will be assigned to a new flow:

$$f'_i = \frac{T'_i}{T} = \beta_E \frac{1}{N} + \beta_C \frac{T_i}{T} + \beta_L \frac{T_i^2}{R_i \sum_j (T_j^2 / R_j)}. \tag{3}$$

While (3) describes a very broad design space, we provide some intuition on how each component acts through a simple example, which is described in full in section 4.1. Consider a network balancing traffic to a given prefix between two paths with equal bottleneck capacity $L_1 = L_2 = 120$Mbps. For the duration of the experiment, a source sends traffic according to the same distribution. At $t = 300$s, cross-traffic is generated towards client C through L_1. At $t = 600$s, a greater quantity cross-traffic is generated towards client C through L_2. The results for this example are shown in figure 2 in four different guises, and in each case we show the throughput, loss rate and the proportion of flows assigned to each path as computed by the PREFLEX balancer.

We first investigate the effect of equalisation, which is necessary to ensure all paths are continually probed. Equalisation alone however leads to an inefficient use of the network if there is a mismatch in path capacity, as is clearly visible in figure 2(a) as congestion arises on either link. Such behaviour arises in traditional traffic engineering, which resorts to equalising traffic weighted by a local link's capacity. Where a bottleneck is remote and distinct however, such behaviour will lead traffic across all links to be roughly bound by the capacity of the slowest path, as can be seen between $t = 300$s and $t = 600$s where throughput over the first path is dragged down by congestion on a second path.

Figure 2(b) and 2(c) on the other hand show distinctive behaviour when using a solely conservative or solely loss-driven approach. Predictably for small values of loss the loss-driven approach overreacts and flaps between either path. While the net effect of these oscillations does not result in significant losses, a conservative approach lends itself more naturally to situations where loss is too small to provide a reliable indicator on path quality. Once loss becomes significant however a conservative approach is unable to drive traffic in order to balance loss across both paths. More worryingly, a pure conservative approach demonstrates the wrong dynamics, as highlighted between $t = 250$s and $t = 300$s. Despite being saturated, the balancer continues to push more traffic towards path 1 while the second path remains under-utilized, dropping its throughput further.

Note that in figure 2 the conservative approach obtains higher throughput than its loss-driven counterpart once loss settles in. While this may seem advantageous, in reality we shall show that this behaviour is detrimental to the system as a whole and only by balancing loss can social welfare be optimized [15].

3.3 Balancing between conservative and loss-driven

We wish to tune PREFLEX to balance between conservative and loss-driven modes for differing regimes of loss. Equalisation is required in some measure as every path must attract some traffic if it is not to fall out of use ([15], remark 2). If this is not the case, a path with relatively high loss will never be probed to determine whether it has improved. The remaining two components must be adjusted to be able to respond adequately to loss while not being overly sensitive to statistically insignificant fluctuations in loss. We define γ to replace both β_C and β_L in (3) and adjust between conservative and loss-driven modes:

$$f_i' = \beta_E \frac{1}{N} + (1-\beta_E)\left(\gamma \frac{T_i}{T} + (1-\gamma)\frac{T_i^2}{R_i \sum_j (T_j^2/R_j)}\right) \tag{4}$$

As a result, β_E may now vary between $[0,1]$. A simple but effective value for γ is to define a minimum average loss μ_{min} below which we do not react to loss, and react increasingly as the average loss μ becomes more significant:

$$\gamma = \begin{cases} \frac{\mu_{min}}{\mu}, & \text{if } \mu > \mu_{min} \\ 1, & \text{if } \mu \leq \mu_{min} \end{cases} \tag{5}$$

This completes the PREFLEX balancer, as shown in figure 2(d). The evolution of γ for the example shown in figure 2(d) is shown in figure 3(left). The value of μ_{min} was set to 0.005.

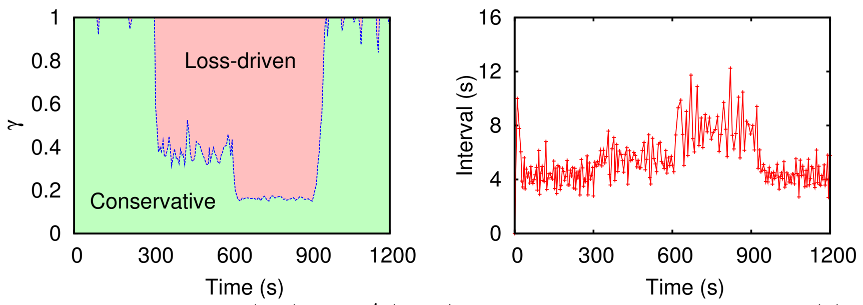


Fig. 3. Parameters γ (left) and τ' (right) as seen in in example in figure 2(d).

3.4 Tuning update interval

Another issue is how often to update the flow split considering the sparseness of loss. Assume that for a given prefix packet loss is measured over a given time period τ. It would be useful to tune this τ per prefix in order that the loss estimate was "accurate". If τ is short then only a very small number of packets will be lost. On the other hand, if τ is long then the control system will be unable to react quickly to changes. The idea is to set τ sufficiently small that an accurate measure of loss can be obtained. It is useful therefore, to have a rough estimate of a time period over which it is necessary to measure in order that estimates of loss are accurate to a given degree. Because this time period of measurement is per prefix, this must to some extent take account of how important a given measurement is to the system as a whole (for example, not slowing down measurements because one route with a tiny amount of traffic has an inaccurate measurement). This will be achieved with the concept of a weighted coefficient of variation.

Let t_i be the number of packets transmitted down path i in the time period τ and let l_i be the number of packets which were lost in this time period. (To prevent divide by zero issue set $l_i = 1$ if no packets are lost).

Let p_i be the probability that a given packet is lost on path i and assume that packet loss is a Bernoulli process. An unbiased estimate of p_i is $\hat{p_i} = l_i/t_i$ (it is important to what follows that $\hat{p_i}$ is only an estimate of p_i by "chance" more or fewer packets may have been lost). If packet loss is Bernoulli then l_i has a binomial distribution and its variance σ^2 is given by $t_i p(1-p)$. The coefficient of variation (CV), is a dimensionless measure given by the standard deviation over the mean $c_v = \sigma/\mu$. Keeping the coefficient of variation within some bound δ is a measure of the amount by which an estimate is likely to vary from the true mean. (Note, however, that this is not technically a confidence interval.)

For the number of lost packets on a given prefix i the estimated CV is $\hat{c_v(i)} = \sqrt{t_i \hat{p_i}(1-\hat{p_i})}/t_i\hat{p_i}$. Let r_i be the rate of packet arrival per unit time on i giving $t_i = r_i\tau$. Define W the CV weighted by transmitted packets over the prefix as $W = \sum_i t_i / t\hat{c_v(i)}$ where $t = \sum_i t_i$ and this expands as

$$W = \sum_i \frac{t_i}{t}\sqrt{\frac{(1-\hat{p_i})}{r_i\tau}}.$$

The "accuracy" of the measurement of p_i is determined by the accuracy of l_i and hence, for the prefix as a whole by the CV W. The aim now is to pick the time period for the next measurement τ' such that $W \leq \delta$ for some δ. Assuming that the loss rates and traffic rates will be the same in the next time period will give a good indication of how to set τ'. Therefore for the next time period

$$\delta \geq W = \frac{1}{\sqrt{\tau'}}\sum_i \frac{t_i}{t}\sqrt{\frac{(1-\hat{p_i})}{r_i}}.$$

This gives an estimated minimum time period to set for the next time period. In order to get weighted CV of packet loss (and hence loss rate) equal to or below δ the time period τ' is bounded by

$$\tau' \geq \tau\left(\frac{1}{\delta t}\sum_i \sqrt{t_i - l_i}\right)^2.$$

This equation gives the smallest value to set the time period of measurement to in order that the weighted coefficient of variation of the loss measurement is a given δ.

While a number of simplifying assumptions have been made, such as modelling loss as Bernoulli, the time scale choice is not a critical system parameter so long as it provides "good enough" estimates of loss. The evolution of τ' for the example in figure 2(d) is plotted in figure 3. As throughput decreases, the time inbetween updates is inflated to adjust to the lower occurrence of loss events.

4 Performance Analysis

We evaluate PREFLEX through simulation in ns-3 [16]. Since PREFLEX balances traffic using loss rather than load, there is a need to emulate the end-to-end behaviour of traffic. This proves more challenging than analysis of existing traffic engineering proposals which typically only focus on adjusting load, since we wish to verify the impact of PREFLEX on end-user metrics.

4.1 Methodology

For all simulations we will use the topology displayed in figure 4. The topology links a client domain C to a server domain S through N paths with equal bottlenecks L_i, and total bandwidth $B = \sum L_i$. While a domain is represented as a single entity in figure 4, each domain is composed by a traffic generator connected to a router. Client C generates G simultaneous HTTP-like requests (or "gets") from S according to a specified distribution, described at the end of this section. As traffic flows from S to C, the router within S is responsible for balancing traffic over all available paths.

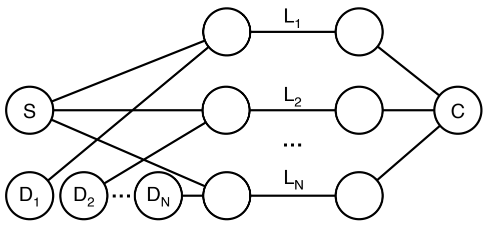


Fig. 4. Simulation topology

Across simulations, as the number of paths increases, total bandwidth B and the number of simultaneous requests G is fixed. In this manner we wish to analyze how PREFLEX balances traffic as the granularity with which it can split traffic becomes coarser.

Since we are interested in evaluating how PREFLEX shifts traffic in response to loss, we introduce additional "dummy" servers D_i which are connected to C through a single path. We partition the total simulation time T into $N+2$ intervals starting on s_i, in which s_0 and s_{N+1} have no traffic to D_i. Starting at time s_i, client C generates g_i requests to D_i according to the same distribution as used to server S. All requests to D_i end at time s_{N+1}. Equation (6) sets the start time s_i for requests to D_i as a function of total simulation time T

and number of paths N. Likewise, equation (7) sets the number of simultaneous requests g_i to D_i as a function of G, the total number of requests to S, and N.

$$s_i = T\frac{i}{N+2} \tag{6}$$

$$\theta_i = \frac{\frac{1}{N+1-i}}{\sum \frac{1}{N+1-i}}, g_i = G\theta_i. \tag{7}$$

Figure 5 illustrates the number of simultaneous gets from C to D_i for $N = 2$ (used in the example shown in figure 2) and $N = 4$. Generating cross-traffic in this manner serves two purposes. Firstly, $\sum g_i = G$, so independently of the number of concurrent paths, the maximum load in the system is $2G$. However, as the number of paths increases, the fluctuation in load for each path becomes smaller, and so we will stress the sensitivity with which PREFLEX balances traffic. Secondly, the number of requests for each D_i over time is the same. Over timescale T, equalisation appears to be an acceptable strategy, however within each interval we will show it performs poorly achieve consistent behaviour. This is a fundamental limitation of offline traffic engineering, which is calculated over very long timescales and is unable to adapt as traffic routinely shifts.

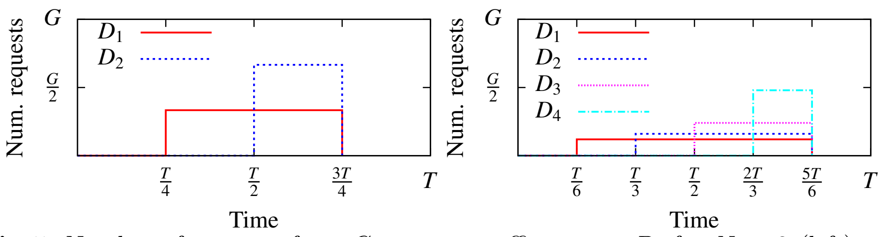


Fig. 5. Number of requests from C to cross traffic servers D_i for $N = 2$ (left) and $N = 4$ (right)

We now specify the settings common to all simulations, including those previously shown in figure 2. Total simulation time T is set to 1200 seconds, while total bandwidth B is fixed at 240Mbps. The number of requests G sent from C to S is set to 240. Upon completing, a request is respawned after an idle period following an exponential distribution with a 15s mean. Transfer size follows a Weibull distribution with an average value of 2MB. These values attempt to reflect traffic to a single prefix with a file size that mimics the small but bursty nature of web traffic, which does not lend itself to being balanced by the end-host. PREFLEX is configured with $\beta_E = 0.05$, $\mu_{min} = 0.01/N$ and $\delta = 0.005$.

4.2 Varying bottleneck distribution

We start by examining the case where all bottlenecks share the same bandwidth, $L_i = B/N$, and compare PREFLEX to equalisation, which mimics traffic engineering techniques based on hashing flow tuples and assigning them to a path. The goodput, calculated as the total data transfered to client C by flows completed within T, is shown for both equalisation and PREFLEX methods in figure 6. While both saturate most available bandwidth, equalisation leads to disproportionate distribution of goodput amongst competing traffic. As loss is not equalised over all paths, the amount of goodput achieved by servers D_i differs despite demand being similar.

Equalisation, even when weighted according to local link capacity, is often prone to remote bottlenecks. We investigate the effect of differing bottlenecks by repeating previous simulations with the same total bandwidth B, but with L_i set proportionally to B in a similar manner to (7), that is $L_i = \theta_i B$.

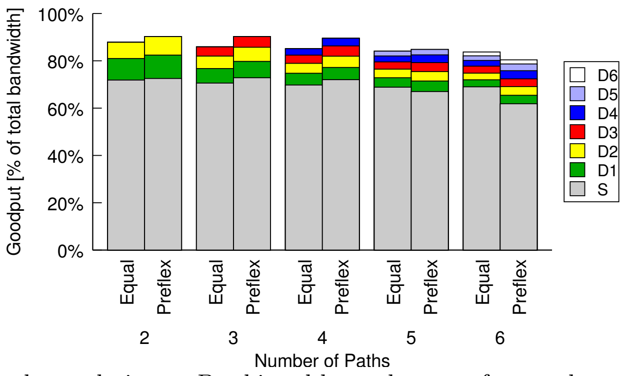


Fig. 6. Goodput relative to B achieved by each server for equal capacity links.

Figure 6 shows the goodput as a proportion of total link bandwidth for the case where all links have equal bandwidth. We vary the number of links N, and for each case compare equalisation (as illustrated in 2(a)) and PREFLEX as the balancing methods used. The bulk of goodput originates from server S, which is the only domain to be connected to all links. If traffic is correctly balanced, we expect to see servers D_{1-N} generate the same amount of goodput.

In this scenario, equalisation can be seen as the optimal static TE solution, yet both approaches bear similar performance. With no knowledge of topology, link bandwidth or expected traffic matrices, PREFLEX is able to adequately mimic the performance of the static TE solution for the case where such an approach is best suited.

Where bottleneck bandwidth is unequal however equalisation proves inadequate. Once again comparing goodput (figure 7) we highlight two significant shortcomings of equalisation which PREFLEX overcomes. Firstly, goodput for S drops as N increases. Unable to realize it is overloading a path, equalisation is reduced to sending traffic over each link at approximately the same rate as the most congested link. In contrast, PREFLEX detects congestion and adapts accordingly. Secondly, the incorrect distribution of traffic due to equalisation in S distorts the goodput of others servers. While in PREFLEX goodput from D_{1-N} is perfectly balanced, with equalisation traffic crossing the most congested links are directly affected by another domain's inability to distribute its traffic appropriately. It may seem unfair to judge equalisation for cases where there

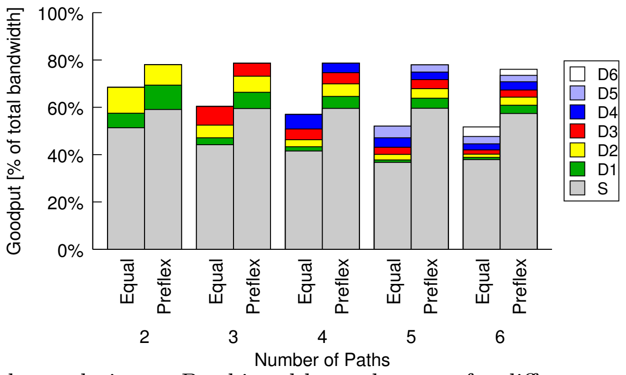


Fig. 7. Goodput relative to B achieved by each server for different capacity links.

is a mismatch in link capacity, however this mismatch between link weight and path capacity arises regularly as operators continue to adjust traffic engineering according to local conditions, with little thought spared for the impact this may have further downstream.

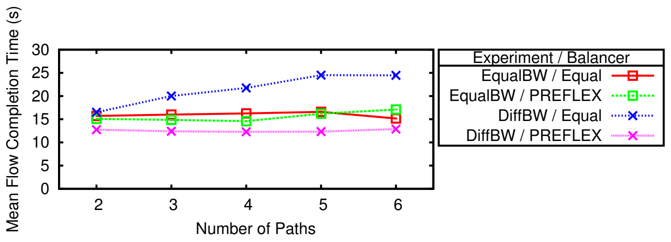


Fig. 8. Mean average flow completion time for equal and differing bottleneck links.

This impact is in turn perceived by users, who experience longer flow completion times, as shown in figure 8. In the equal bandwidth case the flow completion time is similar for both balancers. Where bandwidth differs however, PREFLEX outperforms equalisation and maintains a stable performance when balancing over all six paths. This shows that the algorithm scales well as the number of available paths increases.

5 Conclusions and further work

In this paper we have introduced congestion balancing using PREFLEX. PREFLEX uses packet marking to estimate and balance loss across multiple paths. It requires no per-flow state or significant changes at routers, is computationally simple to implement, and does not cause packet reordering.

PREFLEX has been implemented and evaluated in ns-3 for dynamic traffic scenarios where it balances traffic using different strategies which are weighted according to the network conditions it detects. In conditions where loss is deemed significant, PREFLEX balances congestion between paths. In the absence of sustained loss PREFLEX assigns traffic based on current throughput. By balancing between these strategies PREFLEX can operate in a variety of dynamic traffic settings and has been shown to perform as well as the ideal static traffic assignment bandwidths are equal. Where bandwidth asymmetry arises, PREFLEX successfully balances loss with no significant degradation of performance as both the number of paths and inherent complexity of balancing increases.

By readjusting traffic according to end-to-end metrics, PREFLEX is unique in proposing congestion, rather than just load, as an essential metric for traffic engineering and signals a novel approach in bridging the divide between traffic engineering and congestion control.

References

1. Fortz, B., Thorup, M.: Optimizing OSPF/IS-IS weights in a changing world. Selected Areas in Communications, IEEE Journal on DOI - 10.1109/JSAC.2002.1003042 **20**(4) (2002) 756–767
2. Wang, Y., Wang, Z.: Explicit routing algorithms for internet traffic engineering. Computer Communications and Networks, 1999. Proceedings. Eight International Conference on (1999) 582–588
3. Feamster, N., Borkenhagen, J., Rexford, J.: Guidelines for interdomain traffic engineering. SIGCOMM Computer Communication Review **33**(5) (Oct 2003)
4. Gao, L., Rexford, J.: Stable internet routing without global coordination. Networking, IEEE/ACM Transactions on **9**(6) (2001) 681–692
5. Quoitin, B., Pelsser, C., Swinnen, L., Bonaventure, O., Uhlig, S.: Interdomain traffic engineering with BGP. Communications Magazine **41**(5) (2003) 122– 128
6. Wang, N., Ho, K., Pavlou, G., Howarth, M.: An overview of routing optimization for internet traffic engineering. IEEE Communications Surveys & Tutorials **10**(1) (2008) 36–56
7. Wischik, D., Handley, M., Braun, M.: The resource pooling principle. ACM SIGCOMM Computer Communication Review **38**(5) (2008) 47–52
8. Elwalid, A., Jin, C., Low, S., Widjaja, I.: MATE: multipath adaptive traffic engineering. Computer Networks **40**(6) (2002) 695–709
9. Kandula, S., Katabi, D., Davie, B., Charny, A.: Walking the tightrope: Responsive yet stable traffic engineering. ACM SIGCOMM Computer Communication Review **35**(4) (2005) 264
10. Thaler, D.: Evolution of the ip model. Internet Draft draft-iab-ip-model-evolution-02.txt, IETF. (Work in progress). (2010)
11. Araújo, J.T., Rio, M., Pavlou, G.: A mutualistic resource pooling architecture. Third Workshop on Re-Architecting the Internet (Re-Arch), Philadelphia (2010)
12. Sinha, S., Kandula, S., Katabi, D.: Harnessing TCP's burstiness with flowlet switching. 3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets) (2004)
13. Briscoe, B., Jacquet, A., Cairano-Gilfedder, C.D., Salvatori, A., Soppera, A., Koyabe, M.: Policing congestion response in an internetwork using re-feedback. ACM SIGCOMM Computer Communication Review **35**(4) (2005) 288
14. Kohler, E., Handley, M., Floyd, S.: Designing dccp: Congestion control without reliability. ACM SIGCOMM Computer Communication Review **36**(4) (2006) 38
15. Kelly, F., Voice, T.: Stability of end-to-end algorithms for joint routing and rate control. ACM SIGCOMM Computer Communication Review **35**(2) (2005) 12
16. Network Simulator 3. http://www.nsnam.org